REDES DE COMPUTADORES

5ª EDIÇÃO

TANENBAUM | WETHERALL

REDES DE COMPUTADORES

5ª EDIÇÃO

Tradução

Daniel Vieira

Revisão Técnica

Prof. Dr. Isaias Lima

Professor com dedicação exclusiva à
Universidade Federal de Itajubá e dos cursos de pós-graduação em
Engenharia Elétrica e Ciência e Tecnologia da Computação.

Tradução autorizada a partir da edição original, em inglês, *Computer networks,* 5th edition, publicada pela Pearson Education, Inc., sob o selo Prentice Hall.

Diretor editorial: Roger Trimer
Gerente editorial: Sabrina Cairo
Editor de aquisição: Vinícius Souza
Coordenadora de produção editorial: Thelma Babaoka
Editora de texto: Arlete Sousa
Preparação: Renata Campos
Revisão: Alberto Bononi e Célia Arruda
Capa: Celso Blanes sobre projeto original de Linda Knowles e Susan Paradise e ilustração de Jason Consalvo
Editoração eletrônica e diagramação: Globaltec Editorial & Marketing

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

Tanenbaum, Andrew S.
Redes de computadores / Andrew S. Tanenbaum e David Wetherall ; tradução Daniel Vieira ; revisão técnica Isaías Lima. -- São Paulo : Pearson Prentice Hall, 2011.

Título original: Computer networks
5. ed. americana
ISBN 978-85-7605-924-0

1. Redes de computadores I. Wetherall, David. II. Título. III. Série.

10-14066 CDD-004.6

Índice para catálogo sistemático:
1. Redes de computadores :
Comunicação de dados 004.6

2ª reimpressão – junho 2013
Direitos exclusivos para a língua portuguesa cedidos à
Pearson Education do Brasil Ltda.,
uma empresa do grupo Pearson Education
Rua Nelson Francisco, 26
CEP 02712-100 – São Paulo – SP – Brasil
Fone: (11) 2178-8686 – Fax: (11) 2178-8688
vendas@pearson.com

Para Suzanne, Barbara, Daniel, Aron, Marvin, Matilde,
e à memória de Bram e Sweetie π *(AST)*

Para Katrin, Lucy e Pepper (DJW)

Sumário

Prefácio

Este livro está agora em sua quinta edição. Cada edição correspondeu a uma fase diferente no modo como as redes de computadores eram usadas. Quando surgiu a primeira edição, em 1980, as redes eram uma curiosidade acadêmica. Quando a segunda edição foi publicada, em 1988, as redes eram usadas por universidades e grandes empresas. Quando a terceira edição apareceu, em 1996, as redes de computadores, especialmente a Internet, já tinham se tornado uma realidade no dia a dia de milhões de pessoas. Com a quarta edição, em 2003, as redes sem fios e os computadores móveis já haviam se tornado comuns para acessar a Web e a Internet. Agora, na quinta edição, as redes tratam da distribuição de conteúdo (especialmente vídeos usando CDNs e redes peer-to-peer) e os telefones móveis são pequenos computadores na Internet.

O que há de novo na quinta edição

Entre as muitas mudanças neste livro, a mais importante é o acréscimo do professor David J. Wetherall como coautor. David traz um rico conhecimento de redes, tendo trabalhado exaustivamente projetando redes metropolitanas há mais de 20 anos. Ele tem trabalhado com Internet e redes sem fios desde essa época e é professor na Universidade de Washington, onde tem ensinado e feito pesquisa sobre redes de computadores e assuntos relacionados durante a última década.

Naturalmente, o livro também tem muitas mudanças para acompanhar o mundo em constante mutação das redes de computadores. Entre elas, estão materiais revisados e novos sobre:

- redes sem fios (802.12 e 802.16);
- as redes 3G usadas por telefones inteligentes (smartphones);
- RFID e redes de sensores;
- distribuição de conteúdo usando CDNs;
- redes peer-to-peer;
- mídia em tempo real (armazenadas, streaming e ao vivo);
- telefonia na Internet (Voice over IP);
- redes tolerantes a atrasos.

A seguir, apresentamos uma lista detalhada, capítulo por capítulo.

O Capítulo 1 tem a mesma função introdutória da quarta edição, mas o conteúdo foi revisado e atualizado. A Internet, redes de telefonia móvel, 802.11, RFID e redes de sensores são discutidos como exemplos de redes de computadores. O material original sobre Ethernet — com suas extensões — foi removido, além do material sobre ATM.

O Capítulo 2, que abrange a camada física, expandiu o estudo da modulação digital (incluindo o OFDM, bastante usado nas redes sem fios) e redes 3G (baseadas em CDMA). Novas tecnologias são discutidas, incluindo Fiber to the Home (FTTH) e redes por linhas de energia elétrica.

O Capítulo 3, sobre enlaces ponto a ponto, foi melhorado de duas maneiras. O material sobre códigos para detecção e correção de erros foi atualizado e também inclui uma breve descrição dos códigos modernos que são importantes na prática (por exemplo, códigos convolucionais e LDPC). Agora, os exemplos de protocolos usam Pacotes sobre SONET e ADSL. Infelizmente, o material sobre verificação de protocolo foi retirado, pois é pouco utilizado.

No Capítulo 4, sobre a subcamada MAC, os princípios não mudam, mas as tecnologias mudaram. As seções sobre as redes-exemplo foram devidamente refeitas, incluindo gigabit Ethernet, 802.11, 802.16, Bluetooth e RFID. A explicação sobre comutação de LAN, incluindo as VLANs, também foi atualizada.

O Capítulo 5, sobre a camada de rede, abrange o mesmo terreno da quarta edição. As revisões foram a atualização do material e o acréscimo em profundidade, principalmente para a qualidade de serviço (relevante para mídia em tempo real) e rede interligada. As seções sobre BGP, OSPF e CIDR foram expandidas, assim como o tratamento do roteamento multicast. O roteamento anycast foi incluído nesta edição.

O Capítulo 6, sobre a camada de transporte, teve materiais acrescentados, revisados e removidos. O material novo descreve a rede tolerante a atrasos e o controle de congestionamento em geral. O material revisado atualiza e expande a abordagem do controle de congestionamento TCP. O material removido descrevia o modelo em camadas da rede e serviço orientado a conexões, algo que agora raramente é visto.

O Capítulo 7, sobre aplicações, também foi atualizado e ampliado. Embora o material sobre DNS e correio eletrô-

nico seja semelhante ao da quarta edição, nos últimos anos aconteceram muitos desenvolvimentos no uso da Web, streaming de mídia e entrega de conteúdo. Consequentemente, as seções sobre Web e streaming de mídia foram atualizadas. Uma nova seção inclui distribuição de conteúdo, incluindo CDNs e redes peer-to-peer.

O Capítulo 8, sobre segurança, ainda abrange a criptografia de chave simétrica e pública para confidencialidade e autenticidade. O material sobre as técnicas usadas na prática, incluindo firewalls e VPNs, foi atualizado, com acréscimo de novo material sobre segurança 802.11 e Kerberos V5.

O Capítulo 9 contém uma lista renovada de leituras sugeridas e uma bibliografia abrangente, com mais de 300 citações sobre a literatura atual. Mais da metade delas são citações de artigos e livros escritos a partir de 2000, e o restante são citações de artigos clássicos.

Lista de acrônimos

Os livros de informática estão repletos de acrônimos. Este não é uma exceção. Ao concluir a leitura deste volume, todas estas siglas terão um sentido claro para você: ADSL, AES, AJAX, AODV, AP, ARP, ARQ, AS, BGP, BOC, CDMA, CDN, CGI, CIDR, CRL, CSMA, CSS, DCT, DES, DHCP, DHT, DIFS, DMCA, DMT, DMZ, DNS, DOCSIS, DOM, DSLAM, DTN, FCFS, FDD, FDDI, FDM, FEC, FIFO, FSK, FTP, GPRS, GSM, HDTV, HFC, HMAC, HTTP, IAB, ICANN, ICMP, IDEA, IETF, IMAP, IMP, IP, IPTV, IRTF, ISO, ISP, ITU, JPEG, JSP, JVM, LAN, LATA, LEC, LEO, LLC, LSR, LTE, MAN, MFJ, MIME, MPEG, MPLS, MSC, MTSO, MTU, NAP, NAT, NRZ, NSAP, OFDM, OSI, OSPF, PAWS, PCM, PGP, PIM, PKI, POP, POTS, PPP, PSTN, QAM, QPSK, RED, RFC, RFID, RPC, RSA, RTSP, SHA, SIP, SMTP, SNR, SOAP, SONET, SPE, SSL, TCP, TDD, TDM, TSAP, UDP, UMTS, URL, VLAN, VSAT, WAN, WDM e XML. Mas não se preocupe. Cada um aparecerá em **negrito** e será cuidadosamente definido antes de ser usado. Apenas como teste, veja quantos você consegue identificar *antes* de ler o livro; escreva o número na margem e depois tente novamente *ao término* da obra.

Como usar o livro

Para ajudar os instrutores a usar este livro para cursos variando de trimestres a semestres, estruturamos os capítulos em básico e opcional. As seções marcadas com um "*" no sumário são as opcionais. Se uma seção principal (por exemplo, 2.7) for marcada dessa maneira, todas as suas seções são opcionais. Elas oferecem material sobre tecnologias de rede, que é útil, mas que pode ser omitido em um curso mais rápido, sem perda de continuidade. Naturalmente, os alunos deverão ser encorajados a ler essas seções também, desde que tenham tempo, uma vez que todo o material está atualizado e é valioso.

O Companion Website desta obra (www.pearson.com.br/tanenbaum), traz, para professores, Manual de soluções e apresentações em PowerPoint.

Agradecimentos

Muitas pessoas nos ajudaram durante o curso da quinta edição. Gostaríamos de agradecer especialmente a Emmanuel Agu (Worcester Polytechnic Institute), Yoris Au (University of Texas em Antonio), Nikhil Bhargava (Aircom International, Inc.), Michael Buettner (University of Washington), John Day (Boston University), Kevin Fall (Intel Labs), Ronald Fulle (Rochester Institute of Technology), Ben Greenstein (Intel Labs), Daniel Halperin (University of Washington), Bob Kinicki (Worcester Polytechnic Institute), Tadayoshi Kohno (University of Washington), Sarvish Kulkarni (Villanova University), Hank Levy (University of Washington), Ratul Mahajan (Microsoft Research), Craig Partridge (BBN), Michael Piatek (University of Washington), Joshua Smith (Intel Labs), Neil Spring (University of Maryland), David Teneyuca (University of Texas em Antonio), Tammy VanDegrift (University of Portland) e Bo Yuan (Rochester Institute of Technology), por oferecerem ideias e sugestões. Melody Kadenko e Julie Svendsen proporcionaram apoio administrativo a David.

Shivakant Mishra (University of Colorado em Boulder) e Paul Nagin (Chimborazo Publishing, Inc.) pensaram em muitos problemas de fim de capítulo novos e desafiadores. Nossa editora na Pearson, Tracy Dunkelberger, novamente foi útil de muitas maneiras, grandes e pequenas. Melinda Haggerty e Jeff Holcomb fizeram um bom trabalho mantendo as coisas em ordem. Steve Armstrong (LeTourneau University) preparou os slides do PowerPoint. Stephen Turner (University of Michigan em Flint) revisou brilhantemente os recursos Web e os simuladores que acompanham o texto. Nossa revisora de texto, Rachel Head, é uma combinação estranha: ela tem o olho de uma águia e a memória de um elefante. Depois de ler todas as suas correções, nos perguntamos como conseguimos passar da terceira série (do ensino fundamental!).

Finalmente, chegamos às pessoas mais importantes. Suzanne já passou por isso 19 vezes, e ainda tem paciência e amor sem fim. Barbara e Marvin agora sabem a diferença entre livros bons e ruins, e sempre são minha inspiração para a produção de livros bons. Daniel e Matilde são bem-vindos à nossa família. Aron provavelmente não lerá este livro tão cedo, mas ele gosta das figuras das zebras no Capítulo 8 (AST). Katrin e Lucy ofereceram apoio sem fim e sempre conseguiram manter um sorriso em meu rosto. Obrigado (DJW).

ANDREW S. TANENBAUM
DAVID J. WETHERALL

Capítulo 1

Introdução

Cada um dos três séculos anteriores foi dominado por uma única nova tecnologia. O século XVIII foi a época dos grandes sistemas mecânicos que acompanharam a Revolução Industrial. O século XIX foi a era das máquinas a vapor. As principais conquistas tecnológicas do século XX se deram no campo da aquisição, do processamento e da distribuição de informações. Entre outros desenvolvimentos, vimos a instalação das redes de telefonia em escala mundial, a invenção do rádio e da televisão, o nascimento e o crescimento sem precedentes da indústria da informática, o lançamento dos satélites de comunicação e, naturalmente, a Internet.

Como resultado do rápido progresso tecnológico, essas áreas estão convergindo rapidamente no século XXI e as diferenças entre coleta, transporte, armazenamento e processamento de informações estão desaparecendo rapidamente. Organizações com centenas de escritórios dispersos por uma extensa área geográfica normalmente esperam, com um simples pressionar de um botão, poder examinar o status atual até mesmo de suas filiais mais remotas. À medida que cresce nossa capacidade de colher, processar e distribuir informações, torna-se ainda maior a demanda por formas mais sofisticadas de processamento de informação.

Apesar de a indústria de informática ainda ser jovem em comparação a outros setores (por exemplo, o de automóveis e o de transportes aéreos), foi simplesmente espetacular o progresso que os computadores experimentaram em um curto período. Durante as duas primeiras décadas de sua existência, os sistemas computacionais eram altamente centralizados, em geral instalados em uma grande sala, muitas vezes com paredes de vidro, através das quais os visitantes podiam contemplar, embevecidos, aquela grande maravilha eletrônica. Uma empresa de médio porte ou uma universidade contava apenas com um ou dois computadores, enquanto as grandes instituições tinham, no máximo, algumas dezenas. Era pura ficção científica a ideia de que, em quarenta anos, computadores muito mais poderosos, menores que selos postais, seriam produzidos em massa, aos bilhões.

A fusão dos computadores e comunicações teve uma profunda influência na forma como os sistemas computacionais são organizados. O conceito então dominante de 'centro de computação' como uma sala com um grande computador ao qual os usuários levam seu trabalho para processamento agora está completamente obsoleto (embora os centros de dados com milhares de servidores de Internet estejam se tornando comuns). O velho modelo de um único computador atendendo a todas as necessidades computacionais da organização foi substituído por outro em que os trabalhos são realizados por um grande número de computadores separados, porém interconectados. Esses sistemas são chamados **redes de computadores**. O projeto e a organização dessas redes são os temas deste livro.

Ao longo do livro, utilizaremos a expressão 'rede de computadores' para indicar um conjunto de computadores autônomos interconectados por uma única tecnologia. Dois computadores estão interconectados quando podem trocar informações. A conexão não precisa ser feita por um fio de cobre; também podem ser usadas fibras ópticas, micro-ondas, ondas de infravermelho e satélites de comunicações. Existem redes de muitos tamanhos, modelos e formas, como veremos mais adiante. Elas normalmente estão conectadas para criar redes maiores, com a **Internet** sendo o exemplo mais conhecido de uma rede de redes.

Há uma certa confusão na literatura quanto à distinção entre uma rede de computadores e um **sistema distribuído**. A principal diferença entre eles é que, em um sistema distribuído, um conjunto de computadores independentes parece ser, para os usuários, um único sistema coerente. Em geral, ele tem um único modelo ou paradigma apresentado aos usuários. Com frequência, uma camada de software sobre o sistema operacional, chamada **middleware**, é responsável pela implementação desse modelo. Um exemplo bem conhecido de sistema distribuído é a **World Wide Web**. Ela trabalha em cima da Internet e apresenta um modelo em que tudo tem a aparência de um documento (página Web).

Em uma rede de computadores, essa coerência, esse modelo e esse software estão ausentes. Os usuários ficam expostos às máquinas reais, sem qualquer tentativa, por parte do sistema, de fazer as máquinas parecerem e atuarem de modo coerente. Se as máquinas tiverem hardware e sistemas operacionais distintos, isso será totalmente visível para os usuários. Se um usuário quiser executar um programa em uma máquina remota, ele terá de efetuar o logon nessa máquina e executar o programa lá.

Na prática, um sistema distribuído é um sistema de software instalado na parte superior de uma rede. O software dá ao sistema um alto grau de coesão e transparência. Consequentemente, é o software (e em particular o sistema

operacional) que determina a diferença entre uma rede e um sistema distribuído, não o hardware.

Apesar disso, há uma considerável sobreposição entre os dois assuntos. Por exemplo, os sistemas distribuídos e as redes de computadores precisam movimentar arquivos. A diferença está em quem chama a movimentação: o sistema ou o usuário. Embora este livro seja dedicado principalmente a redes, muitos dos assuntos também são importantes em sistemas distribuídos. Para obter mais informações sobre sistemas distribuídos, consulte Tanenbaum e Van Steen (2007).

1.1 Usos de redes de computadores

Antes de começarmos a examinar detalhadamente as questões técnicas, vale a pena dedicar algum tempo para explicar por que as pessoas estão interessadas em redes de computadores e com que finalidade essas redes podem ser usadas. Afinal, se ninguém estivesse interessado em redes de computadores, poucas teriam sido montadas. Começaremos com os usos tradicionais em empresas, e depois passaremos para as redes domésticas e aos desenvolvimentos mais recentes relacionados a usuários móveis, terminando com as questões sociais.

1.1.1 Aplicações comerciais

Muitas empresas têm um número significativo de computadores. Por exemplo, uma empresa pode ter um computador para cada trabalhador e os usa para projetar produtos, escrever documentos e elaborar a folha de pagamento. Inicialmente, alguns desses computadores podem funcionar isoladamente dos outros, mas, em determinado momento, a gerência pode decidir conectá-los para extrair e correlacionar informações sobre a empresa inteira.

Em termos um pouco mais genéricos, a questão aqui é o **compartilhamento de recursos**. O objetivo é deixar todos os programas, equipamentos e, especialmente, dados ao alcance de todas as pessoas na rede, independentemente da localização física do recurso ou do usuário. Um exemplo óbvio e bastante disseminado é um grupo de funcionários de um escritório que compartilham uma impressora comum. Nenhum dos indivíduos realmente necessita de uma impressora privativa, e uma impressora de grande capacidade conectada em rede muitas vezes é mais econômica, mais rápida e de manutenção mais fácil que um grande conjunto de impressoras individuais.

Porém, talvez mais importante que compartilhar recursos físicos, como impressoras e unidades de fita, seja compartilhar informações. Toda empresa, grande ou pequena, tem uma dependência vital de informações computadorizadas. A maioria das empresas tem registros de clientes, informações de produtos, estoques, extratos financeiros, informações sobre impostos e muitas outras informações on-line. Se todos os computadores de um banco sofressem uma pane, ele provavelmente não duraria mais que cinco minutos. Uma instalação industrial moderna, com uma linha de montagem controlada por computadores, não duraria nem cinco segundos. Hoje, até mesmo uma pequena agência de viagens ou uma empresa jurídica com três pessoas depende intensamente de redes de computadores para permitir a seus funcionários acessar informações e documentos relevantes de forma quase instantânea.

Para empresas menores, todos os computadores provavelmente se encontram em um único escritório ou talvez em um único prédio; porém, no caso de empresas maiores, os computadores e funcionários podem estar dispersos por dezenas de escritórios e instalações em muitos países. Apesar disso, um vendedor em Nova York às vezes precisa acessar um banco de dados de estoque de produtos localizado em Cingapura. Redes chamadas **VPNs** (**Virtual Private Networks**) podem ser usadas para unir as redes individuais em diferentes locais em uma rede estendida. Em outras palavras, o mero fato de um usuário estar a 15 mil quilômetros de distância de seus dados não deve impedi-lo de usá-los como se eles fossem dados locais. Resumindo, trata-se de uma tentativa de dar fim à 'tirania da geografia'.

No mais simples dos termos, é possível imaginar que o sistema de informações de uma empresa consista em um ou mais bancos de dados com informações da empresa e algum número de funcionários que necessitem acessá-los remotamente. Nesse modelo, os dados são armazenados em poderosos computadores chamados **servidores**. Normalmente, eles são instalados e mantidos em um local central por um administrador de sistemas. Ao contrário, os funcionários têm em suas mesas máquinas mais simples, chamadas **clientes**, com as quais acessam dados remotos, por exemplo, para incluir em planilhas eletrônicas que estão elaborando. (Algumas vezes, faremos referência ao usuário humano da máquina cliente como o 'cliente', mas deve ficar claro, pelo contexto, se estamos nos referindo ao computador ou a seu usuário.) As máquinas cliente e servidor são conectadas entre si por uma rede, como ilustra a Figura 1.1. Observe que mostramos a rede como uma simples elipse, sem qualquer detalhe. Utilizaremos essa forma quando mencionarmos uma rede no sentido mais abstrato. Quando forem necessários mais detalhes, eles serão fornecidos.

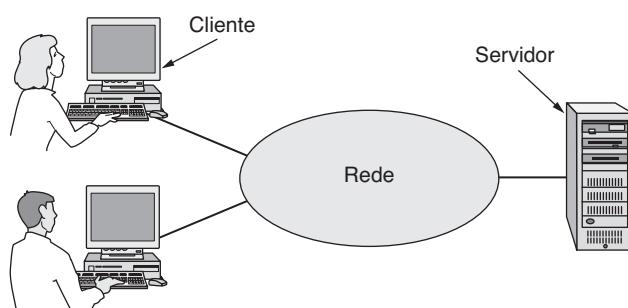



Figura 1.1 | Uma rede com dois clientes e um servidor.

Todo esse arranjo é chamado **modelo cliente-servidor**. Ele é bastante usado e forma a base de grande parte do uso da rede. A realização mais popular é a de uma **aplicação Web**, em que o servidor fornece páginas Web com base em seu banco de dados em resposta às solicitações do cliente. O modelo cliente-servidor é aplicável quando cliente e servidor estão, ambos, no mesmo prédio (e pertencem à mesma empresa), mas também quando estão muito afastados. Por exemplo, quando uma pessoa em casa acessa uma página na World Wide Web, o mesmo modelo é empregado, com o servidor Web remoto fazendo o papel do servidor e o computador pessoal do usuário sendo o cliente. Sob a maioria das condições, um único servidor pode lidar com um grande número (centenas ou milhares) de clientes simultaneamente.

Se examinarmos o modelo cliente-servidor em detalhes, veremos que dois processos (por exemplo, programas em execução) são envolvidos, um na máquina cliente e um na máquina servidora. A comunicação toma a forma do processo cliente enviando uma mensagem pela rede ao processo servidor. Então, o processo cliente espera por uma mensagem de resposta. Quando o processo servidor recebe a solicitação, ele executa o trabalho solicitado ou procura pelos dados solicitados e envia uma resposta de volta. Essas mensagens são mostradas na Figura 1.2.

Um segundo objetivo da configuração de uma rede de computadores está relacionado às pessoas, e não às informações ou mesmo aos computadores. Uma rede de computadores pode oferecer um poderoso **meio de comunicação** entre os funcionários. Praticamente toda empresa com dois ou mais computadores tem o recurso de **e-mail** (**correio eletrônico**), que os funcionários utilizam de forma geral para suprir uma grande parte da comunicação diária. De fato, os funcionários trocam mensagens de e-mail sobre os assuntos mais corriqueiros, mas grande parte das mensagens com que as pessoas lidam diariamente não tem nenhum significado, porque os chefes descobriram que podem enviar a mesma mensagem (normalmente, sem conteúdo) a todos os seus subordinados, bastando pressionar um botão.

Ligações telefônicas entre os funcionários podem ser feitas pela rede de computadores, em vez de pela companhia telefônica. Essa tecnologia se chama **telefonia IP** ou **Voice over IP** (**VoIP**) quando a tecnologia da Internet é empregada. O microfone e o alto-falante em cada extremo podem pertencer a um telefone habilitado para VoIP ou ao computador do funcionário. As empresas descobriram que essa é uma forma maravilhosa de economizar nas contas telefônicas.

Outras formas de comunicação mais ricas são possíveis com as redes de computadores. O vídeo pode ser acrescentado ao áudio, de modo que os funcionários em locais distantes possam ver e ouvir um ao outro enquanto realizam uma reunião. Essa técnica é uma ferramenta eficiente para eliminar o custo e o tempo anteriormente dedicado às viagens. O **compartilhamento de desktop** permite que trabalhadores remotos vejam e interajam com uma tela de computador. Com isso, duas ou mais pessoas em locais distantes podem participar de uma reunião, vendo e ouvindo uns aos outros e até mesmo escrevendo um relatório em um quadro-negro compartilhado. Quando um funcionário faz uma mudança em um documento on-line, os outros podem ver a mudança imediatamente, em vez de esperar vários dias por uma carta. Essa agilidade facilita a cooperação entre grupos de pessoas dispersas, enquanto anteriormente isso era impossível. Atualmente, estão começando a ser usadas outras formas de coordenação remota mais ambiciosas, como a telemedicina (por exemplo, no monitoramento de pacientes remotos), mas elas podem se tornar muito mais importantes. Algumas vezes, diz-se que a comunicação e o transporte estão disputando uma corrida, e a tecnologia que vencer tornará a outra obsoleta.

Um terceiro objetivo para muitas empresas é realizar negócios eletronicamente, em especial com clientes e fornecedores. Esse novo modelo é chamado de **e-commerce** (**comércio eletrônico**) e tem crescido rapidamente nos últimos anos. Empresas aéreas, livrarias e outros varejistas descobriram que muitos clientes gostam da conveniência de fazer compras em casa. Consequentemente, muitas empresas oferecem catálogos de seus produtos e serviços e recebem pedidos on-line. Fabricantes de automóveis, aeronaves e computadores, entre outros, compram subsistemas de diversos fornecedores e depois montam as peças. Utilizando redes de computadores, os fabricantes podem emitir pedidos eletronicamente, conforme o necessário. Isso reduz a necessidade de grandes estoques e aumenta a eficiência.

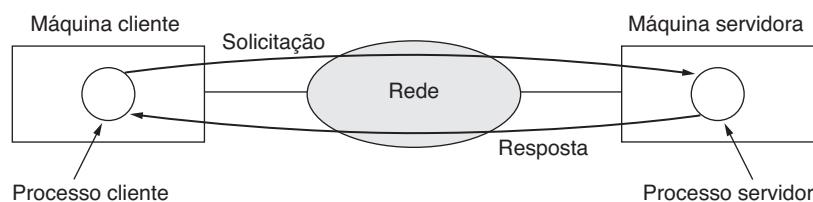



Figura 1.2 | O modelo cliente-servidor envolve solicitações e respostas.

1.1.2 Aplicações domésticas

Em 1977, Ken Olsen era presidente da Digital Equipment Corporation, então o segundo maior fornecedor de computadores de todo o mundo (depois da IBM). Quando lhe perguntaram por que a Digital não estava seguindo a tendência do mercado de computadores pessoais, ele disse: 'Não há nenhuma razão para qualquer indivíduo ter um computador em casa'. A história mostrou o contrário, e a Digital não existe mais. As pessoas inicialmente compravam computadores para processamento de textos e jogos. Nos últimos anos, talvez a maior motivação seja o acesso à Internet. Agora, muitos dispositivos eletrônicos de consumo, como conversores digitais, consoles de jogos e rádio-relógios, já vêm com computadores e redes de computadores embutidas, especialmente redes sem fio, e as redes domésticas são bastante usadas para entretenimento, incluindo escuta, exibição e criação de música, fotos e vídeos.

O acesso à Internet oferece, aos usuários domésticos, **conectividade** a computadores remotos. Assim como as empresas, os usuários domésticos podem acessar informações, comunicar-se com outras pessoas e comprar produtos e serviços com o comércio eletrônico. O principal benefício agora vem da conexão com o exterior da casa. Bob Metcalfe, o inventor da Ethernet, formulou a hipótese de que o valor de uma rede é proporcional ao quadrado do número de usuários, pois esse é aproximadamente o número de conexões diferentes que podem ser feitas (Gilder, 1993). Essa hipótese é conhecida como a 'lei de Metcalfe'. Ela ajuda a explicar como a tremenda popularidade da Internet vem de seu tamanho.

O acesso a informações remotas tem várias formas. Ele pode significar navegar na World Wide Web para obter informações ou apenas por diversão. As informações disponíveis incluem artes, negócios, culinária, governo, saúde, história, passatempos, recreação, ciência, esportes, viagens e muitos outros. A diversão surge sob tantas formas que fica difícil mencioná-las, e também se apresenta em outras formas que é melhor não serem mencionadas.

Muitos jornais são publicados on-line e podem ser personalizados. Por exemplo, às vezes é possível solicitar todas as informações sobre políticos corruptos, grandes incêndios, escândalos envolvendo celebridades e epidemias, mas dispensar qualquer notícia sobre esportes. Algumas vezes, até mesmo é possível transferir os artigos selecionados por download para o disco rígido enquanto você dorme. À medida que essa tendência continuar, ela causará desemprego maciço entre os jovens entregadores de jornais, mas as empresas jornalísticas gostam disso, porque a distribuição sempre foi o elo mais fraco na cadeia de produção inteira. Naturalmente, para que esse modelo funcione, primeiro eles terão de descobrir como ganhar dinheiro nesse novo mundo, algo que não é totalmente óbvio, pois os usuários da Internet esperam que tudo seja de graça.

A próxima etapa, além de jornais (e de revistas e publicações cientificas), é a biblioteca digital on-line. Muitas organizações profissionais, como a ACM (www.acm.org) e a IEEE Computer Society (www.computer.org), já têm muitas publicações e anais de conferências on-line. Leitores de e-books (livros eletrônicos) e bibliotecas on-line podem tornar os livros impressos obsoletos. Os céticos devem observar o efeito que a máquina de impressão teve sobre os manuscritos medievais com iluminuras.

Grande parte dessa informação é acessada por meio do modelo cliente-servidor, mas existe um modelo diferente, popular, para acessar informações, que recebe o nome de comunicação **peer-to-peer** (ou não hierárquica) (Parameswaran et al., 2001). Nessa forma de comunicação, indivíduos que constituem um grupo livre podem se comunicar com outros participantes do grupo, como mostra a Figura 1.3. Em princípio, toda pessoa pode se comunicar com uma ou mais pessoas; não existe qualquer divisão estrita entre clientes e servidores.

Muitos sistemas peer-to-peer, como BitTorrent (Cohen, 2003), não possuem qualquer banco de dados de conteúdo central. Em vez disso, cada usuário mantém seu próprio banco de dados no local e oferece uma lista de

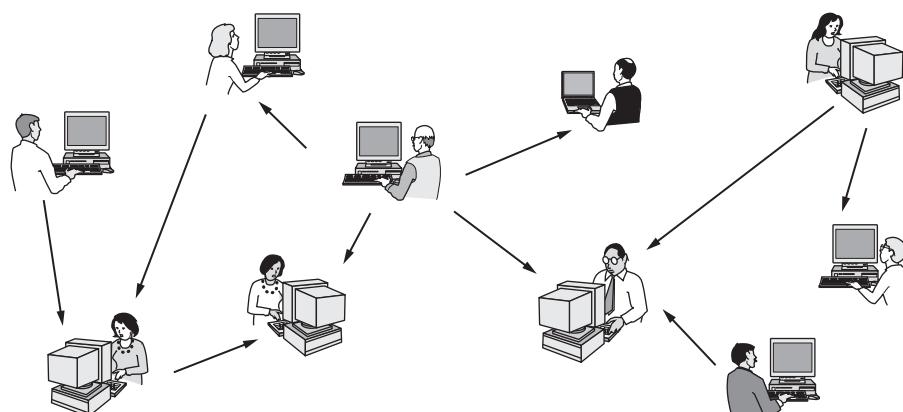



Figura 1.3 | Em um sistema não hierárquico não existem clientes e servidores estritos.

outras pessoas vizinhas que são membros do sistema. Um novo usuário pode, então, ir até qualquer membro existente para ver o que ele tem e obter os nomes de outros membros para inspecionar mais conteúdo e mais nomes. Esse processo de pesquisa pode ser repetido indefinidamente, até criar um grande banco de dados local do que existe no sistema inteiro. Essa é uma atividade que seria tediosa para as pessoas, mas os computadores podem sobressair nisso.

A comunicação peer-to-peer normalmente é usada para compartilhar música e vídeos. Ela realmente alcançou o auge por volta de 2000, com um serviço de compartilhamento de música chamado Napster, que foi encerrado depois daquilo que provavelmente foi a maior violação de direitos autorais em toda a história registrada (Lam e Tan, 2001; Macedonia, 2000). Também existem aplicações legais para a comunicação peer-to-peer. Entre elas estão fãs compartilhando músicas de domínio público, famílias compartilhando fotos e filmes, e usuários baixando pacotes de software públicos. Na verdade, uma das aplicações mais populares de toda a Internet, o correio eletrônico, é basicamente peer-to-peer. Essa forma de comunicação provavelmente crescerá bastante no futuro.

Todas as aplicações anteriores envolvem interações entre uma pessoa e um banco de dados remoto, repleto de informações. A segunda grande categoria de utilização de redes é a comunicação entre pessoas, basicamente a resposta do século XXI ao telefone do século XIX. O correio eletrônico (e-mail) já é usado diariamente por milhões de pessoas em todo o mundo e seu uso está crescendo rapidamente. Em geral, ele já contém áudio e vídeo, além de texto e imagens. O odor pode demorar um pouco mais.

Hoje em dia, qualquer adolescente é fanático pela **troca de mensagens instantâneas**. Esse recurso, derivado do programa *talk* do UNIX, em uso desde aproximadamente 1970, permite que duas pessoas digitem mensagens uma para a outra em tempo real. Também existem serviços de mensagens para várias pessoas, como o serviço **Twitter**, permitindo que as pessoas enviem pequenas mensagens de texto, chamadas 'tweets', para seu círculo de amigos ou outras audiências interessadas.

A Internet pode ser usada pelas aplicações para transportar áudio (por exemplo, estações de rádio pela Internet) e vídeo (por exemplo, o YouTube). Além de ser um modo barato de se comunicar com amigos distantes, essas aplicações podem oferecer experiências ricas, como teleaprendizado, o que significa assistir a aulas às 8 da manhã sem a inconveniência de ter de levantar da cama. Com o passar do tempo, o uso das redes para melhorar a comunicação entre os seres humanos poderá ser mais importante do que qualquer outro. Isso pode se tornar extremamente importante para pessoas que estão geograficamente distantes, dando-lhes o mesmo acesso aos serviços que as pessoas morando no meio de um grande centro urbano já têm.

Entre as comunicações interpessoais e o acesso à informação estão as aplicações de **rede social**. Aqui, o fluxo de informações é controlado pelos relacionamentos que as pessoas declaram umas às outras. Uma das redes sociais mais populares é o **Facebook**. Ela permite que as pessoas atualizem seus perfis pessoais e compartilhem as atualizações com outras pessoas de quem elas declararam ser amigas. Outras aplicações de rede social podem fazer apresentações por meio de amigos dos amigos, enviar mensagens de notícias aos amigos, como o Twitter, e muito mais.

Ainda de forma mais livre, os grupos de pessoas podem trabalhar juntos para criar conteúdo. Uma **wiki**, por exemplo, é um website colaborativo que os membros de uma comunidade editam. A wiki mais famosa é a **Wikipedia**, uma enciclopédia que qualquer um pode editar, mas existem milhares de outras wikis.

Nossa terceira categoria é o comércio eletrônico no sentido mais amplo do termo. A atividade de fazer compras em casa já é popular e permite ao usuário examinar catálogos on-line de milhares de empresas. Alguns desses catálogos são interativos, mostrando produtos de diferentes pontos de vista e em configurações que podem ser personalizadas. Depois que um cliente compra um produto eletronicamente, se ele não souber como usá-lo, o suporte técnico on-line poderá ser consultado.

Outra área em que o comércio eletrônico já é uma realidade é o acesso a instituições financeiras. Muitas pessoas já pagam suas contas, administram contas bancárias e manipulam seus investimentos eletronicamente. Sem dúvida, essa tendência crescerá à medida que as redes se tornarem mais seguras.

Uma área que praticamente ninguém previu é a de brechós eletrônicos (e-brechó?). Leilões on-line de objetos de segunda mão tornaram-se uma indústria próspera. Diferente do comércio eletrônico tradicional, que segue o modelo cliente-servidor, os leilões on-line são um tipo de sistema peer-to-peer, no sentido de que os consumidores podem atuar como compradores e vendedores.

Algumas dessas formas de comércio eletrônico utilizam pequenas abreviações baseadas no fato de que 'to' e '2' têm a mesma pronúncia em inglês. As mais populares estão relacionadas na Tabela 1.1.

Nossa quarta categoria é o entretenimento. Ela tem feito grande progresso nas residências ao longo dos últimos anos, com a distribuição de música, programas de rádio e televisão, e os filmes pela Internet começando a competir com os meios tradicionais. Os usuários podem localizar, comprar e baixar músicas em MP3 e filmes com qualidade de DVD e depois incluí-los em sua coleção pessoal. Os programas da TV agora alcançam muitos lares via sistemas de **IPTV** (**IP TeleVision**), que são baseados na tecnologia IP, em vez das transmissões de TV a cabo ou rádio. As aplicações de mídia permitem que os usuários sintonizem estações de rádio pela Internet ou assistam a episódios recentes

Abreviação	Nome completo	Exemplo
B2C	Business-to-consumer	Pedidos de livros on-line
B2B	Business-to-business	Fabricante de automóveis solicitando pneus a um fornecedor
G2C	Government-to-consumer	Governo distribuindo eletronicamente formulários de impostos
C2C	Consumer-to-consumer	Leilões on-line de produtos usados
P2P	Peer-to-peer	Compartilhamento de música

Tabela 1.1 | Algumas formas de comércio eletrônico.

de seus programas favoritos na TV. Naturalmente, todo esse conteúdo pode ser passado aos diferentes aparelhos, monitores e alto-falantes de sua casa, normalmente com uma rede sem fio.

Logo, talvez seja possível selecionar qualquer filme ou programa de televisão, qualquer que seja a época ou país em que tenha sido produzido, e exibi-lo em sua tela no mesmo instante. Novos filmes poderão se tornar interativos e, ocasionalmente, o usuário poderá ser solicitado a interferir no roteiro (Macbeth deve matar Duncan ou esperar pelo momento certo?), com cenários alternativos para todas as hipóteses. A televisão ao vivo também poderá se tornar interativa, com os telespectadores participando de programas de perguntas e respostas, escolhendo entre concorrentes e assim por diante.

Outra forma de entretenimento são os jogos eletrônicos. Já temos jogos de simulação em tempo real com vários participantes, como os de esconde-esconde em um labirinto virtual, e simuladores de voo em que os jogadores de uma equipe tentam abater os jogadores da equipe adversária. Os mundos virtuais oferecem um ambiente persistente, em que milhares de usuários podem experimentar uma realidade compartilhada com gráficos tridimensionais.

Nossa última categoria é a **computação ubíqua**, em que a computação está embutida no dia a dia, como na visão de Mark Weiser (1991). Muitos lares já estão preparados com sistemas de segurança que incluem sensores em portas e janelas, e existem muitos outros sensores que podem ser embutidos em um monitor doméstico inteligente, como no consumo de energia. Seus medidores de eletricidade, gás e água também poderiam informar o uso pela rede. Isso economizaria dinheiro, pois não seria preciso enviar funcionários para ler a medição do consumo. E seus detectores de fumaça poderiam ligar para os bombeiros em vez de fazer muito barulho (o que não adianta muito se não houver alguém em casa). À medida que o custo dos sensores e da comunicação diminui, mais e mais aplicações de medição e envio de informações serão disponibilizadas pelas redes.

Cada vez mais, os dispositivos eletrônicos de consumo estão em rede. Por exemplo, algumas câmeras de última geração já possuem capacidade para rede sem fio, utilizada para enviar fotos a um monitor próximo, para serem exibidas. Fotógrafos profissionais de esportes também podem enviar fotos para seus editores em tempo real, primeiro sem fio, para um ponto de acesso, e em seguida para a Internet. Dispositivos como televisores que se conectam na tomada da parede podem usar a **rede de energia elétrica** para enviar informações pela casa, nos fios que transportam eletricidade. Pode não ser surpresa ter esses objetos na rede, mas objetos que não imaginamos como computadores também podem detectar e comunicar informações. Por exemplo, seu chuveiro poderá registrar o uso de água, dando-lhe um *feedback* visual enquanto você se ensaboa, e informar para uma aplicação de monitoramento ambiental doméstico quando tiver terminado, para ajudá-lo a economizar em sua conta de água.

Uma tecnologia chamada **RFID** (**Radio Frequency IDentification**) levará essa ideia ainda mais adiante no futuro. Etiquetas RFID são chips passivos (ou seja, não têm bateria), do tamanho de um selo, e já podem ser afixados em livros, passaportes, animais de estimação, cartões de crédito e outros itens em casa e fora dela. Isso permite que os leitores de RFID localizem e se comuniquem com os itens por uma distância de até vários metros, dependendo do tipo de RFID. Originalmente, a RFID foi comercializada para substituir os códigos de barras. Ela ainda não teve muito sucesso porque os códigos de barras são gratuitos e as etiquetas de RFID custam alguns centavos. Naturalmente, as etiquetas de RFID oferecem muito mais e seu preço está caindo rapidamente. Elas podem transformar o mundo real na Internet de coisas (ITU, 2005).

1.1.3 Usuários móveis

Computadores móveis, como notebooks e computadores portáteis, constituem um dos segmentos de mais rápido crescimento no setor de informática. Suas vendas já superaram as de computadores desktop. Por que alguém desejaria um? As pessoas que estão em trânsito normalmente desejam usar seus dispositivos móveis para ler e enviar e-mail, 'tuitar', assistir a filmes, baixar música, jogar ou simplesmente navegar pelas informações na Web. Elas querem fazer todas as coisas que fazem em casa e no es-

critório. Naturalmente, elas querem fazê-las em qualquer lugar, na terra, no mar ou no ar.

A **conectividade** à Internet habilita muitos desses usos móveis. Como ter uma conexão com fios é impossível nos carros, barcos e aviões, há muito interesse nas redes sem fio. As redes celulares operadas pelas empresas de telefonia são um tipo conhecido de rede sem fio que dá cobertura para telefones móveis. Os **hotspots** sem fio baseados no padrão 802.11 são outro tipo de rede sem fio para computadores móveis. Eles surgem em todo lugar a que as pessoas vão, resultando em uma malha com cobertura em cafés, hotéis, aeroportos, escolas, trens e aviões. Qualquer um com um laptop e um modem sem fio pode simplesmente ligar seu computador e estar conectado à Internet pelo hotspot, como se o computador estivesse conectado a uma rede com fios.

As redes sem fio têm grande valor para frotas de caminhões, táxis, veículos de entrega e funcionários de serviços de assistência técnica, que precisam manter-se em contato com sua base de operações. Por exemplo, em muitas cidades, os motoristas de táxi são homens de negócios independentes, em vez de serem funcionários de uma empresa de táxi. Em algumas dessas cidades, os táxis têm uma tela de vídeo que o motorista pode observar. Ao receber uma chamada de cliente, um despachante central digita os pontos de partida e destino. Essa informação é exibida nas telas de vídeo dos motoristas e um aviso sonoro é emitido. O primeiro motorista a pressionar um botão na tela de vídeo obtém a corrida.

As redes sem fios também são importantes para os militares. Se, de uma hora para outra, for necessário travar uma guerra em qualquer lugar no mundo, talvez não seja possível contar com a possibilidade de usar a infraestrutura de rede local. Será melhor levar seu próprio equipamento de rede.

Embora as redes sem fios e a computação móvel frequentemente estejam relacionadas, elas não são idênticas, como mostra a Tabela 1.2. Aqui, observamos uma distinção entre redes **sem fio fixas** e **sem fio móveis**. Algumas vezes, até mesmo os notebooks podem estar conectados por fios. Por exemplo, se um viajante conecta um notebook à tomada de rede em um quarto de hotel, ele tem mobilidade sem precisar utilizar uma rede sem fio.

Sem fio	Móvel	Aplicações típicas
Não	Não	Computadores desktop em escritórios
Não	Sim	Um notebook usado em um quarto de hotel
Sim	Não	Redes em edifícios que não dispõem de fiação
Sim	Sim	PDA para registrar o estoque de uma loja

Tabela 1.2 | Combinações de redes sem fio e computação móvel.

Por outro lado, alguns computadores sem fio não são portáteis. Em casa e nos escritórios ou hotéis que não possuem cabeamento adequado, pode ser mais conveniente conectar computadores desktop ou aparelhos sem fio do que instalar os fios. A instalação de uma rede sem fio pode exigir pouco mais do que adquirir uma pequena caixa com alguns componentes eletrônicos, retirá-la da embalagem e conectá-la ao equipamento. Essa solução pode ser muito mais barata do que pedir que um trabalhador monte conduítes para passar fiação no prédio.

Finalmente, também há as verdadeiras aplicações móveis, sem fios, como pessoas percorrendo lojas com um PDA e registrando o estoque. Em muitos aeroportos mais cheios, os funcionários de devolução de carros alugados trabalham no estacionamento com computadores móveis sem fios. Eles leem os códigos de barras ou chips de RFID dos carros devolvidos, e seu dispositivo móvel, que possui uma impressora embutida, chama o computador principal, recebe a informação da locação e imprime a conta no ato.

Talvez o impulso fundamental das aplicações móveis, sem fio, seja o telefone móvel. O envio de **mensagens de texto**, ou **torpedos**, é tremendamente popular. Ele permite que um usuário de telefone móvel digite uma mensagem curta que é, então, entregue pela rede celular para outro assinante móvel. Poucas pessoas teriam previsto, há dez anos, que adolescentes digitando mensagens de texto curtas em telefones móveis seria uma grande forma de ganhar dinheiro para as companhias telefônicas. Mas o envio de texto (ou **Short Message Service**, como é conhecido fora dos Estados Unidos) é muito lucrativo, pois custa à operadora uma pequena fração de um centavo para repassar uma mensagem de texto, um serviço pelo qual elas cobram muito mais.

A tão aguardada convergência entre telefones e a Internet finalmente chegou, e acelerará o crescimento das aplicações móveis. **Smartphones**, como o popular iPhone, combinam aspectos de telefones e computadores móveis. As redes celulares (3G e 4G) às quais eles se conectam podem oferecer serviços de dados rápidos para usar a Internet, bem como para lidar com ligações telefônicas. Muitos telefones mais avançados também se conectam a hotspots sem fio, e automaticamente alternam entre as redes para escolher a melhor opção para o usuário.

Outros aparelhos eletrônicos também podem usar redes celulares e hotspot para permanecer conectados a computadores remotos. Os leitores de livros eletrônicos podem baixar um livro recém-adquirido, a próxima edição de uma revista ou o jornal de hoje, onde quer que eles estejam. Os porta-retratos eletrônicos podem ser atualizados automaticamente com imagens novas.

Como os telefones móveis têm conhecimento de suas localizações, normalmente porque são equipados com receptores de **GPS** (**Global Positioning System**), alguns serviços são intencionalmente dependentes do local. Ma-

pas móveis e orientações são candidatos óbvios, visto que seu telefone habilitado com GPS e seu carro provavelmente têm uma ideia melhor de onde você está do que você mesmo. O mesmo pode acontecer com as buscas por uma livraria próxima ou um restaurante chinês, ou a previsão do tempo local. Outros serviços podem registrar o local, como a anotação de fotos e vídeos com o local em que foram feitos. Essa anotação é conhecida como 'geomarcação'.

Uma área em que esses dispositivos podem se destacar é chamada **m-commerce** (**mobile-commerce**) (Senn, 2000). Mensagens de texto curtas do telefone móvel são usadas para autorizar pagamentos de alimentos em máquinas, ingressos de cinema e outros itens pequenos, ao invés de dinheiro em espécie e cartões de crédito/débito. O débito aparece, então, na conta do telefone celular. Quando equipado com tecnologia **NFC** (**Near Field Communication**), o telefone móvel pode atuar como um smartcard com RFID e interagir com um leitor próximo para realizar o pagamento. A força motriz por trás desse fenômeno consiste em uma mistura de fabricantes de PDAs sem fios e operadores de redes que estão tentando descobrir como obter uma fatia do comércio eletrônico. Do ponto de vista da loja, esse esquema pode poupar-lhes a maior parte das tarifas da empresa de cartões de crédito, o que pode significar uma porcentagem elevada. É claro que esse plano pode ter efeito contrário ao desejado, pois os clientes de uma loja poderiam usar os leitores de RFID ou código de barras em seus dispositivos móveis para verificar os preços dos concorrentes antes de comprar e, depois, obter instantaneamente um relatório detalhado de onde mais o item poderia ser adquirido na região e a que preço.

Uma enorme vantagem do m-commerce é que os usuários de telefones celulares se acostumaram a pagar por tudo (ao contrário dos usuários da Internet, que esperam conseguir tudo de graça). Se um website cobrasse uma taxa para permitir a seus clientes efetuar pagamentos com cartão de crédito, haveria uma imensa reclamação dos usuários. Se, no entanto, uma operadora de telefonia celular permitisse às pessoas pagar por itens de uma loja usando o telefone e depois cobrassem uma tarifa por essa conveniência, provavelmente isso seria aceito como algo normal. O tempo dirá.

Não há dúvida de que os usos dos computadores móveis e sem fio aumentarão rapidamente no futuro, à medida que o tamanho dos computadores diminui, provavelmente de maneiras que ninguém é capaz de prever. Vejamos algumas das possibilidades. **Redes de sensores** são compostas de nós que colhem e repassam, sem fios, as informações que eles detectam sobre o estado do mundo físico. Os nós podem fazer parte de itens familiares, como carros ou telefones, ou então podem ser pequenos dispositivos separados. Por exemplo, seu carro poderia colher dados sobre sua localização, velocidade, vibração e economia a partir de seu sistema de diagnóstico de bordo e enviar essa informação para um banco de dados (Hull et al., 2006). Esses dados podem ajudar a localizar buracos, planejar viagens evitando estradas congestionadas e informar se você é um 'devorador de combustível', em comparação com outros motoristas no mesmo trecho de estrada.

Redes de sensores estão revolucionando a ciência oferecendo diversos dados sobre o comportamento que não poderiam ser observados anteriormente. Um exemplo é o rastreamento da migração de zebras individuais, colocando um pequeno sensor em cada animal (Juang et al., 2002). Os pesquisadores inseriram um computador sem fio em um cubo de 1 mm de borda (Warneke et al., 2001). Com computadores móveis desse tamanho, até mesmo pássaros, roedores e insetos podem ser rastreados.

Até mesmo usos rotineiros, como em parquímetros, podem ser significativos, pois utilizam dados que anteriormente não estavam disponíveis. Os parquímetros sem fio podem aceitar pagamentos com cartão de crédito ou débito, com verificação instantânea pelo enlace sem fio. Eles também podem relatar quando estão em uso pela rede sem fio. Isso permite aos motoristas baixar um mapa de estacionamento atual para seu carro, de modo que podem encontrar um ponto disponível mais facilmente. É claro que, quando um parquímetro expira, ele também pode verificar a presença de um carro (emitindo um sinal a partir dele) e informar a expiração ao agente de estacionamento. Estima-se que os municípios dos Estados Unidos poderiam coletar mais de US$ 10 bilhões apenas dessa maneira (Harte et al., 2000).

Computadores embarcados são outra aplicação promissora. Relógios inteligentes com rádios fazem parte de nosso espaço mental desde seu aparecimento nas tiras de quadrinhos de Dick Tracy em 1946; agora você pode comprá-los. Outros dispositivos desse tipo podem ser implantados, como marca-passos e bombas de insulina. Alguns deles podem ser controlados por uma rede sem fio. Isso permite que os médicos os testem e reconfigurem mais facilmente. Isso também poderia levar a alguns problemas desagradáveis se os dispositivos forem tão seguros como um PC comum, e puderem ser facilmente adulterados (Halperin et al., 2008).

1.1.4 Questões sociais

As redes de computadores, assim como a imprensa há cerca de 500 anos, permitem que os cidadãos comuns manifestem suas opiniões de maneiras que não eram possíveis anteriormente. Mas, junto com o lado bom vem o lado ruim, pois essa nova liberdade traz consigo uma série de questões sociais, políticas e éticas. Vamos mencionar rapidamente algumas delas; um estudo completo exigiria um livro inteiro, pelo menos.

Redes sociais, quadros de mensagens, sites de compartilhamento de conteúdo e uma série de outras aplicações permitem que as pessoas compartilhem suas ideias com indivíduos de mesmo pensamento. Desde que os assuntos

sejam restritos a assuntos técnicos ou passatempos como jardinagem, não surgirão muitos problemas.

Os problemas começam a vir à tona quando as pessoas abordam temas com os quais as pessoas realmente se preocupam, como política, religião ou sexo. Os pontos de vista divulgados podem ser altamente ofensivos para algumas pessoas. Pior ainda, eles podem não ser politicamente corretos. Além disso, as opiniões não estão obrigatoriamente limitadas ao texto; fotos coloridas de alta resolução e mesmo pequenos videoclipes podem ser facilmente compartilhados pelas redes de computadores. Algumas pessoas adotam a visão de que cada um sabe o que faz, mas outras acham que a publicação de certos tipos de materiais (por exemplo, ataques a determinados países ou religiões, pornografia etc.) é simplesmente inaceitável e tem de ser censurada. Diferentes países têm leis distintas e conflitantes sobre esse assunto. Assim, essa polêmica fica cada vez mais acirrada.

No passado, as pessoas abriram processos contra operadores de redes, partindo do princípio de que, a exemplo do que ocorre com os jornais e revistas, eles têm de assumir a responsabilidade pelo conteúdo do que publicam. A resposta inevitável é que uma rede é como uma companhia telefônica ou uma empresa de correios, e que não se pode esperar que ela censure seus usuários.

Agora, deve ser pouco surpreendente descobrir que alguns operadores de rede bloqueiam o conteúdo por seus próprios motivos. Alguns usuários de aplicações peer-to-peer tiveram seus serviços de rede suspensos porque os operadores de rede não acharam lucrativo transportar as grandes quantidades de tráfego enviadas por essas aplicações. Esses mesmos operadores provavelmente gostariam de tratar diferentes empresas de formas diferentes. Se você é uma empresa grande e paga bem, então receberá um bom serviço, mas se você é um peixe pequeno, receberá um serviço fraco. Os oponentes dessa prática argumentam que o peer-to-peer e outro conteúdo deverá ser tratado da mesma maneira, pois todos são apenas bits para a rede. Esse argumento para as comunicações que não são diferenciadas por seu conteúdo, origem ou por quem está fornecendo o conteúdo é conhecido como **neutralidade da rede** (Wu, 2003). Provavelmente, é seguro dizer que esse debate continuará por algum tempo.

Muitas outras partes estão envolvidas na briga pelo conteúdo. Por exemplo, música e filmes pirateados alimentaram o crescimento maciço das redes peer-to-peer, que não agradaram os proprietários dos direitos autorais, os quais ameaçaram (e às vezes tomaram) ações legais. Atualmente, existem sistemas automatizados que procuram redes peer-to-peer e disparam avisos aos operadores e usuários da rede que são suspeitos de infringir direitos autorais. Nos Estados Unidos, esses avisos são conhecidos como **notas de demolição DMCA** pelo **Digital Millennium Copyright Act**. Essa busca é uma corrida armamentista, pois é difícil apanhar a infração de direitos autorais de forma confiável. Até mesmo sua impressora poderia ser pega por engano como uma culpada (Piatek et al., 2008).

As redes de computadores facilitam muito a comunicação. Elas também tornam fácil bisbilhotar o tráfego para as pessoas que controlam a rede. Isso cria conflitos por questões como direitos dos funcionários *versus* direitos do empregador. Muitas pessoas leem e escrevem e-mail no trabalho. Muitos empregadores reivindicaram o direito de ler e possivelmente censurar as mensagens dos funcionários, incluindo mensagens enviadas de um computador doméstico fora dos horários de trabalho. Nem todos os funcionários concordam com isso, especialmente com a última parte.

Outro tópico importante é a relação entre o governo e os direitos dos cidadãos. O FBI instalou um sistema em muitos provedores de serviços da Internet para bisbilhotar todas as mensagens de correio eletrônico que entram e saem, em busca de fragmentos de interesse. O sistema foi originalmente chamado Carnivore, mas a publicidade ruim fez com que ele fosse renomeado com a sigla aparentemente mais inocente DCS1000 (Blaze e Bellovin, 2000; Sobel, 2001; e Zacks, 2001). No entanto, seu objetivo ainda é espionar milhões de pessoas, na esperança de encontrar informações sobre atividades ilegais. Infelizmente para os espiões, a Quarta Emenda à Constituição dos Estados Unidos proíbe buscas do governo sem um mandado de busca, mas o governo ignora isso com frequência.

Naturalmente, o governo não tem o monopólio das ameaças à privacidade das pessoas. O setor privado também faz sua parte, **traçando perfis** dos usuários. Por exemplo, pequenos arquivos, chamados cookies, que os navegadores da Web armazenam nos computadores dos usuários, permitem que as empresas controlem as atividades desses usuários no ciberespaço e também podem permitir que números de cartões de crédito, números de CPF e outras informações confidenciais vazem pela Internet (Berghel, 2001). As empresas que oferecem serviços baseados na Web podem manter grandes quantidades de informações pessoais sobre seus usuários, permitindo-lhes estudar diretamente as atividades do usuário. Por exemplo, o Google pode ler seu e-mail e mostrar propagandas com base em seus interesses, se você usar seu serviço de e-mail, o **Gmail**.

Uma nova guinada com os dispositivos móveis é a privacidade de local (Beresford e Stajano, 2003). Como parte do processo de oferecer serviços a seu dispositivo móvel, os operadores da rede descobrem onde você está em diferentes momentos do dia. Isso lhes permite rastrear seus movimentos. Eles podem saber qual clube você frequenta e qual centro médico você visita.

As redes de computadores também oferecem o potencial para aumentar a privacidade por meio do envio de mensagens anônimas. Em algumas situações, esse recurso pode ser desejável. Além de impedir que as empresas

descubram seus hábitos, ele proporciona, por exemplo, um meio para alunos, soldados, trabalhadores e cidadãos denunciarem o comportamento ilegal de professores, oficiais, superiores e políticos sem medo de possíveis represálias. Por outro lado, nos Estados Unidos e na maioria dos países democráticos, a lei permite especificamente às pessoas acusadas o direito de se confrontarem com o acusador em juízo, de modo que acusações anônimas não podem ser usadas como evidência.

A Internet torna possível encontrar informações com rapidez, mas uma grande parte dessas informações é incorreta, enganosa ou totalmente errada. Aquele aconselhamento médico que você conseguiu na Internet sobre sua dor no peito pode ter vindo de um ganhador do Prêmio Nobel ou de alguém que abandonou os estudos no ensino médio.

Outras informações constantemente são indesejadas. O lixo de correio eletrônico (spam) se tornou parte de nossa vida, pois os spammers coletam milhões de endereços de e-mail e os pretensos profissionais de marketing podem enviar mensagens geradas pelo computador com um custo muito baixo. A inundação de spam resultante compete com o fluxo de mensagens de pessoas reais. Felizmente, o software de filtragem consegue ler e descartar o spam gerado por outros computadores, com menores ou maiores graus de sucesso.

Outro tipo de conteúdo visa a um comportamento criminoso. As páginas Web e as mensagens de e-mail com conteúdo ativo (basicamente, programas ou macros executados na máquina do receptor) podem conter vírus capazes de devastar seu computador. Eles podem ser usados para roubar suas senhas de conta bancária, ou fazer com que seu computador envie spam como parte de uma **botnet** ou um pool de máquinas infectadas.

O **roubo de identidade** (ou **phishing**) finge estar sendo originado de uma parte confiável — por exemplo, seu banco — para tentar roubar informações confidenciais — por exemplo, números de cartão de crédito. O roubo de identidade está se tornando um problema sério, pois os ladrões coletam informações suficientes sobre a pessoa para obter cartões de crédito e outros documentos em nome da vítima.

Pode ser difícil impedir que os computadores se passem por pessoas na Internet. Esse problema levou ao desenvolvimento de **CAPTCHAs**, em que um computador pede a uma pessoa para resolver uma pequena tarefa de reconhecimento, por exemplo, digitar as letras mostradas em uma imagem distorcida, para mostrar que são humanos (von Ahn, 2001). Esse processo é uma variação do famoso teste de Turing, em que uma pessoa faz perguntas por uma rede para julgar se a entidade que responde é humana.

Muitos desses problemas poderiam ser resolvidos se a indústria de informática levasse a sério a segurança dos computadores. Se todas as mensagens fossem criptografadas e autenticadas, seria mais difícil haver danos. Essa tecnologia está bem estabelecida e será estudada em detalhes no Capítulo 8. O problema é que os fornecedores de hardware e software sabem que a inclusão de recursos de segurança custa dinheiro, e seus clientes não exigem tais recursos. Além disso, um número substancial dos problemas é causado por bugs de software, que ocorrem porque os fornecedores continuam a acrescentar mais e mais recursos a seus programas, o que inevitavelmente significa mais código e, portanto, mais bugs. Um imposto sobre novos recursos poderia ajudar, mas isso talvez dificultasse as vendas em poucos trimestres. Um programa de reembolso por software defeituoso talvez fosse ótimo, exceto pelo fato de levar à bancarrota toda a indústria de software no primeiro ano.

As redes de computadores levantam novos problemas legais quando interagem com leis antigas. Jogos de aposta eletrônicos são um exemplo. Os computadores têm simulado coisas há décadas, logo, por que não simular máquinas caça-níqueis, roletas, apostas de vinte e um e outros equipamentos de jogo? Bem, porque é ilegal em muitos lugares. O problema é que o jogo é legal em muitos outros lugares (Inglaterra, por exemplo) e os donos de cassinos de lá entenderam o potencial para jogar pela Internet. O que acontece se o jogador, o cassino e o servidor estiverem em países diferentes, com leis em conflito? Boa pergunta.

1.2 Hardware de rede

Agora é hora de voltarmos nossa atenção das aplicações e aspectos sociais das redes (a diversão) para as questões técnicas envolvidas no projeto da rede (o trabalho). Não existe nenhuma taxonomia de aceitação geral na qual todas as redes de computadores se encaixam, mas duas dimensões se destacam das demais: a tecnologia de transmissão e a escala. Vamos examinar cada uma delas.

Em termos gerais, há dois tipos de tecnologias de transmissão em uso disseminado nos dias de hoje: enlaces de **broadcast** e enlaces **ponto a ponto**.

Os enlaces ponto a ponto conectam pares de máquinas individuais. Para ir da origem ao destino em uma rede composta de enlaces ponto a ponto, mensagens curtas, chamadas **pacotes** em certos contextos, talvez tenham de visitar primeiro uma ou mais máquinas intermediárias. Como normalmente é possível haver várias rotas de diferentes tamanhos, encontrar boas rotas é algo importante em redes ponto a ponto. A transmissão ponto a ponto com exatamente um transmissor e exatamente um receptor às vezes é chamada de **unicasting**.

Ao contrário, as redes de broadcast têm apenas um canal de comunicação, compartilhado por todas as máquinas da rede; os pacotes enviados por qualquer máquina são recebidos por todas as outras. Um campo de endereço dentro do pacote especifica o destinatário pretendido. Quando re-

cebe um pacote, a máquina processa o campo de endereço. Se o pacote se destinar à máquina receptora, esta o processará; se for destinado a alguma outra máquina, o pacote será simplesmente ignorado.

Uma rede sem fio é um exemplo comum de um enlace de broadcast, com a comunicação compartilhada por uma região de cobertura que depende do canal sem fios e da máquina transmissora. Como uma analogia, imagine uma pessoa em uma sala de reunião, gritando: 'Watson, venha cá. Preciso de você'. Embora o pacote possa ser recebido (ouvido) por muitas pessoas, apenas Watson responderá; os outros simplesmente o ignoram.

Os sistemas de broadcast normalmente também oferecem a possibilidade de endereçamento de um pacote a *todos* os destinos usando um código especial no campo de endereço. Quando um pacote com esse código é transmitido, ele é recebido e processado por cada máquina na rede; não é à toa que esse modo de operação é chamado **broadcasting**. Alguns sistemas de broadcasting também admitem a transmissão para um subconjunto de máquinas, o que se conhece como **multicasting**.

Um critério alternativo para classificar as redes é por escalabilidade. A distância é importante como métrica de classificação, pois diferentes tecnologias são usadas em diferentes escalas.

Na Figura 1.4, classificamos vários sistemas processadores por seu tamanho físico aproximado. Na parte superior encontram-se as redes pessoais, redes destinadas a uma única pessoa. Depois aparecem as redes de maior tamanho, que podem ser divididas em locais, metropolitanas e a longas distâncias, ambas em escala crescente. Finalmente, a conexão de duas ou mais redes é chamada rede interligada. A Internet mundial certamente é o mais conhecido (mas não o único) exemplo de uma rede interligada. Logo, teremos redes interligadas ainda maiores, com a **Internet interplanetária**, que conecta redes no espaço sideral (Burleigh et al., 2003).

Distância do interprocessador	Processadores localizados no mesmo	Exemplo
1 m	Metro quadrado	Área pessoal
10 m	Cômodo	Rede local
100 m	Prédio	
1 km	Campus	
10 km	Cidade	Rede metropolitana
100 km	País	Rede a longas distâncias
1.000 km	Continente	
10.000 km	Planeta	A Internet

Figura 1.4 | Classificação de processadores interconectados por escala.

Neste livro, tratamos das redes em todas essas escalas. Nas próximas seções, temos uma breve introdução ao hardware de rede por escala.

1.2.1 Redes pessoais

As **redes pessoais**, ou **PANs** (**Personal Area Networks**), permitem que dispositivos se comuniquem pelo alcance de uma pessoa. Um exemplo comum é uma rede sem fio que conecta um computador com seus periféricos. Quase todo computador tem monitor, teclado, mouse e impressora conectados. Sem usar tecnologia sem fio, essa conexão deve ser feita com cabos. Tantas pessoas têm dificuldade para encontrar os cabos corretos e encaixá-los nos conectores certos (embora normalmente tenham cores diferentes) que a maioria dos vendedores de computador oferece a opção de enviar um técnico à casa do usuário para fazê-lo. Para ajudar esses usuários, algumas empresas se reuniram para projetar uma rede sem fio de curta distância, chamada **Bluetooth**, para conectar esses componentes sem o uso de fios. A ideia é que, se seu dispositivo tem Bluetooth, então você não precisa de cabos. Você simplesmente os liga e eles funcionam juntos. Para muitas pessoas, essa facilidade de operação é uma grande vantagem.

Na forma mais simples, as redes Bluetooth usam um paradigma mestre-escravo da Figura 1.5. A unidade do sistema (o PC) normalmente é o mestre, falando com o mouse, o teclado etc. como escravos. O mestre diz aos escravos quais endereços usar, quando eles podem transmitir, por quanto tempo, quais frequências eles podem usar e assim por diante.

O Bluetooth também pode ser usado em outros ambientes. Ele normalmente é usado para conectar um fone de ouvido sem cabos, e permite que seu aparelho de música digital se conecte a seu carro simplesmente ao entrar no alcance. Um tipo completamente diferente de rede pessoal é formado quando um dispositivo médico embutido, como um marca-passo, bomba de insulina ou aparelho de audição fala com um controle remoto operado pelo usuário. Discutiremos o Bluetooth com mais detalhes no Capítulo 4.

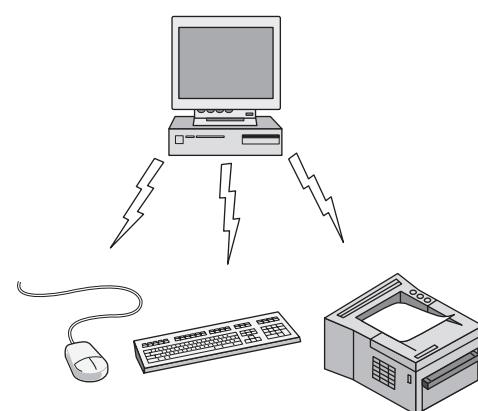



Figura 1.5 | Configuração de rede pessoal Bluetooth.

As redes pessoais também podem ser montadas com outras tecnologias que se comunicam por curtas distâncias, como RFID em smartcards e livros de biblioteca. Estudaremos a RFID no Capítulo 4.

1.2.2 Redes locais

A próxima etapa é a **rede local**, ou **LAN** (**Local Area Network**). Uma LAN é uma rede particular que opera dentro e próximo de um único prédio, como uma residência, um escritório ou uma fábrica. As LANs são muito usadas para conectar computadores pessoais e aparelhos eletrônicos, para permitir que compartilhem recursos (como impressoras) e troquem informações. Quando as LANs são usadas pelas empresas, elas são chamadas **redes empresariais**.

As LANs sem fio são muito populares atualmente, especialmente nas residências, prédios de escritórios mais antigos e outros lugares onde a instalação de cabos é muito trabalhosa. Nesses sistemas, cada computador tem um rádio modem e uma antena, que ele usa para se comunicar com outros computadores. Quase sempre, cada computador fala com um dispositivo no teto, como mostra a Figura 1.6(a). Esse dispositivo, chamado **ponto de acesso** (**AP** – **Access Point**), **roteador sem fio** ou **estação-base**, repassa os pacotes entre os computadores sem fio e também entre eles e a Internet. Ser o AP é como ser o garoto popular na escola, pois todos querem falar com você. Porém, se os outros computadores estiverem próximos o suficiente, eles podem se comunicar diretamente entre si em uma configuração peer-to-peer.

Existe um padrão para as LANs sem fios, chamado **IEEE 802.11**, popularmente conhecido como **WiFi**, que se tornou muito conhecido. Ele trabalha em velocidades de 11 a centenas de Mbps. (Neste livro, vamos aderir à tradição e medir as velocidades de linha em megabits/s, onde 1 Mbps é 1.000.000 bits/s, e gigabits/s, onde 1 Gbps é 1.000.000.000 bits/s.) Discutiremos o padrão 802.11 no Capítulo 4.

As LANs com fios utilizam uma série de tecnologias de transmissão diferentes. A maioria delas usa fios de cobre, mas algumas usam fibra óptica. As LANs são restritas em tamanho, o que significa que o tempo de transmissão, no pior caso, é limitado e conhecido com antecedência. Conhecer esses limites ajuda na tarefa de projetar protocolos de rede. Normalmente, as LANs com fios trabalham em velocidades de 100 Mbps a 1 Gpbs, têm baixo atraso de transporte de dados (microssegundos ou nanossegundos) e com elas ocorrem muito poucos erros. As LANs mais recentes podem operar em até 10 Gbps. Em comparação com as redes sem fios, as LANs com fios as excedem em todas as dimensões de desempenho. É simplesmente mais fácil enviar sinais por um fio ou por uma fibra do que pelo ar.

A topologia de muitas LANs com fios é embutida a partir de enlaces ponto a ponto. O IEEE 802.3, popularmente chamado **Ethernet**, é de longe o tipo mais comum de LAN com fios. A Figura 1.6(b) mostra uma topologia de exemplo da **Ethernet comutada**. Cada computador troca informações usando o protocolo Ethernet e se conecta a um dispositivo de rede chamado **switch**, com um enlace ponto a ponto. Daí o nome. Um switch tem várias **portas**, cada qual podendo se conectar a um computador. A função do switch é repassar os pacotes entre os computadores que estão conectados a ela, usando o endereço em cada pacote para determinar para qual computador enviá-lo.

Para montar LANs maiores, os switches podem ser conectados uns aos outros usando suas portas. O que acontece se você os conectar em um loop? A rede ainda funcionará? Felizmente, os projetistas pensaram nesse caso. É função do protocolo descobrir que caminhos os pacotes devem atravessar para alcançar o computador pretendido com segurança. Veremos como isso funciona no Capítulo 4.

Também é possível dividir uma LAN física grande em duas LANs lógicas menores. Você pode estar se perguntando por que isso seria útil. Às vezes, o layout do equipamento de rede não corresponde à estrutura da organização. Por exemplo, os departamentos de engenharia e finanças de uma empresa poderiam ter computadores na mesma LAN física, pois estão na mesma ala do prédio, mas poderia ser mais fácil administrar o sistema se engenharia e finanças

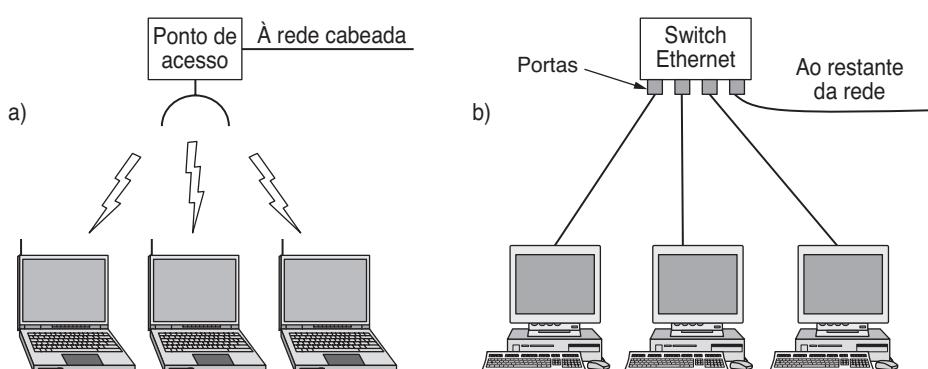



Figura 1.6 | LANs sem fios e cabeadas. (a) 802.11. (b) Ethernet comutada.

tivessem, cada um, sua própria **LAN virtual**, ou **VLAN**. Nesse projeto, cada porta é marcada com uma 'cor', digamos, verde para engenharia e vermelha para finanças. O switch então encaminha pacotes de modo que os computadores conectados às portas verdes sejam separados dos computadores conectados às portas vermelhas. Os pacotes de broadcast enviados em uma porta de cor vermelha, por exemplo, não serão recebidos em uma porta de cor verde, como se existissem duas LANs diferentes. Estudaremos as VLANs no final do Capítulo 4.

Também existem outras topologias de LAN com fios. Na verdade, a Ethernet comutada é uma versão moderna do projeto Ethernet original, que envia todos os pacotes por um único cabo. No máximo uma máquina poderia transmitir com sucesso de cada vez, e um mecanismo distribuído arbitrava o uso e resolvia conflitos da rede compartilhada. Ele usava um algoritmo simples: os computadores poderiam transmitir sempre que o cabo estivesse ocioso. Se dois ou mais pacotes colidissem, cada computador simplesmente esperaria por um tempo aleatório e tentaria mais tarde. Chamaremos essa versão de **Ethernet clássica** para fazer a distinção e, como você já deve imaginar, aprenderá sobre ela no Capítulo 4.

As redes de broadcast, com e sem fios, ainda podem ser divididas em estáticas e dinâmicas, dependendo do modo como o canal é alocado. Em uma alocação estática típica, o tempo seria dividido em intervalos discretos e seria utilizado um algoritmo de rodízio, fazendo com que cada máquina transmitisse apenas no intervalo de que dispõe. A alocação estática desperdiça a capacidade do canal quando uma máquina não tem nada a transmitir durante o intervalo (slot) alocado a ela, e assim a maioria dos sistemas procura alocar o canal dinamicamente (ou seja, por demanda).

Os métodos de alocação dinâmica de um canal comum são centralizados ou descentralizados. No método centralizado de alocação de canal, existe apenas uma entidade, por exemplo, a estação-base nas redes celulares, que determina quem transmitirá em seguida. Para executar essa tarefa, a entidade aceita solicitações e as prioriza de acordo com algum algoritmo interno. No método descentralizado de alocação de canal, não existe nenhuma entidade central; cada máquina deve decidir por si mesma se a transmissão deve ser realizada. Você poderia pensar que isso sempre leva ao caos, mas isso não acontece. Mais tarde, estudaremos muitos algoritmos criados para impedir a instauração do caos potencial.

Vale a pena gastar um pouco mais de tempo discutindo as LANs domésticas. No futuro, é provável que todo dispositivo doméstico seja capaz de se comunicar com cada um dos outros dispositivos, e que todos eles estejam acessíveis pela Internet. Esse é um daqueles conceitos visionários que ninguém solicitou (como os controles remotos de TV ou os telefones celulares) mas, depois que chegaram, ninguém consegue mais imaginar como viver sem eles.

Muitos dispositivos são capazes de se conectar em rede. Entre eles estão computadores, dispositivos de entretenimento como TVs e DVDs, telefones e outros produtos eletrônicos, como câmeras, aparelhos como rádios-relógios e de infraestrutura, como medidores de energia e termostatos. Essa tendência só continuará. Por exemplo, um lar normalmente já tem dezenas de relógios (por exemplo, em aparelhos), todos eles podendo ser ajustados ao horário de verão automaticamente se os relógios estivessem na Internet. O monitoramento remoto da casa é um provável vencedor, pois muitos filhos já crescidos estariam dispostos a gastar algum dinheiro para ajudar seus pais idosos a viverem em segurança em suas próprias casas.

Embora pudéssemos pensar na rede doméstica como apenas outra LAN, ela provavelmente terá propriedades diferentes das outras redes. Primeiro, os dispositivos em rede precisam ser muito fáceis de instalar. Os roteadores sem fio são o item eletrônico de consumo mais devolvido. As pessoas compram um porque querem uma rede sem fio em casa, descobrem que ele não funciona 'conforme sai da caixa' e, depois, o devolvem em vez de escutar aquela musiquinha enquanto esperam na linha pela assistência técnica.

Segundo, a rede e os dispositivos têm de ser à prova de falhas quando em operação. Os condicionadores de ar costumavam ter um único botão com quatro ajustes: DESLIGAR, BAIXO, MÉDIO e ALTO. Agora eles têm manuais de 30 páginas. Uma vez que são ligados em rede, espera-se que apenas o capítulo sobre segurança ocupe 30 páginas. Isso é um problema porque somente usuários de computador estão acostumados a montar produtos que não funcionam; os consumidores de carros, televisores e refrigeradores são muito menos tolerantes. Eles esperam que os produtos funcionem 100 por cento sem precisar contratar um gênio da informática.

Em terceiro lugar, o preço baixo é algo essencial para o sucesso. As pessoas não pagarão US$ 50 a mais por um termostato capaz de se conectar à Internet, porque poucas pessoas consideram importante monitorar a temperatura de sua casa enquanto estão no trabalho. Porém, por US$ 5 a mais, esse acessório teria boa aceitação.

Quarto, deve ser possível começar com um ou dois dispositivos e expandir o alcance da rede gradualmente. Isso significa nenhuma guerra de padrões oferecidos. Dizer aos clientes para comprar periféricos com interfaces IEEE 1394 (FireWire) e alguns anos depois voltar atrás e dizer que USB 2.0 é a interface do mês e depois dizer que 802.11g — opa, não, é melhor 802.11n — quero dizer, 802.16 (diferentes redes sem fio) — deixará os consumidores muito nervosos. A interface de rede terá de permanecer estável por décadas, como os padrões de radiodifusão da televisão.

Quinto, a segurança e a confiabilidade serão muito importantes. Perder alguns arquivos para um vírus de e-mail é uma coisa; permitir que um assaltante desarme seu

sistema de segurança a partir de seu computador móvel e depois saqueie sua casa é algo muito diferente.

Uma questão interessante é saber se as redes domésticas estarão fisicamente conectadas ou se serão redes sem fios. Conveniência e custo favorecem as redes sem fio, pois não existem fios para encaixar, ou ainda, aperfeiçoar. A segurança favorece as redes fisicamente conectadas, pois as ondas de rádio que elas utilizam passam com facilidade pelas paredes. Nem todo mundo fica satisfeito com a ideia de ter os vizinhos pegando carona em sua conexão à Internet e lendo suas mensagens de correio eletrônico. No Capítulo 8, estudaremos como a criptografia pode ser utilizada para proporcionar segurança, mas, no contexto de uma rede doméstica, a segurança tem de ser infalível, mesmo com usuários inexperientes.

Uma terceira opção que pode ser atraente é reutilizar as redes que já estão na residência. O candidato óbvio são os fios elétricos instalados por toda a casa. As **redes de energia elétrica** permitem que os dispositivos conectados às tomadas transmitam informações por toda a casa. De qualquer forma, você já precisa conectar a TV, e dessa forma ela pode obter conectividade com a Internet ao mesmo tempo. A dificuldade é como transportar energia e sinais de dados ao mesmo tempo. Parte da resposta é que eles usam faixas de frequência diferentes.

Resumindo, as LANs domésticas oferecem muitas oportunidades e desafios. A maior parte dos desafios se relaciona à necessidade de que as redes sejam fáceis de administrar, confiáveis e seguras, especialmente nas mãos de usuários não técnicos, além do baixo custo.

1.2.3 Redes metropolitanas

Uma **rede metropolitana**, ou **MAN** (**Metropolitan Area Network**), abrange uma cidade. O exemplo mais conhecido de MANs é a rede de televisão a cabo disponível em muitas cidades. Esses sistemas cresceram a partir de antigos sistemas de antenas comunitárias usadas em áreas com fraca recepção do sinal de televisão pelo ar. Nesses primeiros sistemas, uma grande antena era colocada no alto de colina próxima e o sinal era, então, conduzido até as casas dos assinantes.

Em princípio, esses sistemas eram sistemas *ad hoc* projetados no local. Posteriormente, as empresas começaram a entrar no negócio, obtendo concessões dos governos municipais para conectar cidades inteiras por fios. A etapa seguinte foi a programação de televisão e até mesmo canais inteiros criados apenas para transmissão por cabos. Esses canais costumavam ser bastante especializados, oferecendo apenas notícias, apenas esportes, apenas culinária, apenas jardinagem e assim por diante. Entretanto, desde sua concepção até o final da década de 1990, eles se destinavam somente à recepção de televisão.

A partir do momento em que a Internet atraiu uma audiência de massa, as operadoras de redes de TV a cabo começaram a perceber que, com algumas mudanças no sistema, eles poderiam oferecer serviços da Internet full-duplex (mão dupla) em partes não utilizadas do espectro. Nesse momento, o sistema de TV a cabo começou a se transformar, passando de uma forma de distribuição de televisão para uma rede metropolitana. Em uma primeira aproximação, uma MAN seria semelhante ao sistema mostrado na Figura 1.7. Nessa figura, observamos que os sinais de televisão e de Internet são transmitidos à **central a cabo** centralizada para distribuição subsequente às casas das pessoas. Voltaremos a esse assunto, estudando-o em detalhes no Capítulo 2.

Porém, a televisão a cabo não é a única MAN. Os desenvolvimentos recentes para acesso à Internet de alta velocidade sem fio resultaram em outra MAN, que foi padronizada como IEEE 802.16 e é conhecida popularmente como **WiMAX**. Vamos examiná-la no Capítulo 4.

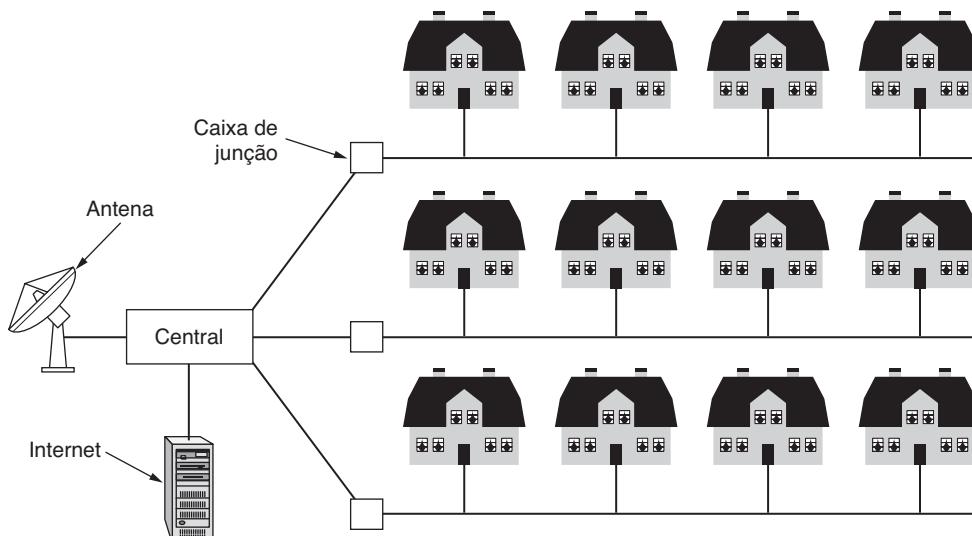



Figura 1.7 | Uma rede metropolitana baseada na TV a cabo.

1.2.4 Redes a longas distâncias

Uma **rede a longa distância**, ou **WAN** (**Wide Area Network**), abrange uma grande área geográfica, com frequência um país ou continente. Vamos começar nossa discussão com as WANs conectadas por fios, usando o exemplo de uma empresa com filiais em diferentes cidades.

A WAN na Figura 1.8 é uma rede que conecta escritórios em Perth, Melbourne e Brisbane. Cada um desses escritórios contém computadores que executam programas (ou seja, aplicações) do usuário. Seguiremos a tradição e chamaremos essas máquinas de **hosts**. O restante da rede que conecta esses hosts é chamada **sub-rede de comunicação** ou, simplificando, apenas **sub-rede**. A tarefa da sub-rede é transportar mensagens de um host para outro, exatamente como o sistema de telefonia transporta as palavras (na realidade, sons) do falante ao ouvinte.

Na maioria das WANs, a sub-rede consiste em dois componentes distintos: linhas de transmissão e elementos de comutação. As **linhas de transmissão** transportam bits entre as máquinas. Elas podem ser formadas por fios de cobre, fibra óptica, ou mesmo enlaces de radiodifusão. A maioria das empresas não tem linhas de transmissão disponíveis, então elas alugam as linhas de uma empresa de telecomunicações. Os **elementos de comutação**, ou apenas comutadores, são computadores especializados que conectam três ou mais linhas de transmissão. Quando os dados chegam a uma interface de entrada, o elemento de comutação deve escolher uma interface de saída para encaminhá-los. Esses computadores de comutação receberam diversos nomes no passado; o nome **roteador** é, agora, o mais comumente usado.

Vale a pena fazermos um breve comentário em relação ao termo 'sub-rede'. Originalmente, seu **único** significado identificava o conjunto de roteadores e linhas de comunicação que transportava pacotes entre os hosts de origem e de destino. Porém, o termo adquiriu um segundo significado, em conjunto com o endereçamento da rede. Discutiremos esse significado no Capítulo 5 e ficaremos com o significado original (uma coleção de linhas de comunicação de dados e roteadores) até chegarmos lá.

A WAN, conforme a descrevemos, é semelhante a uma grande LAN cabeada, mas existem algumas diferenças importantes que vão além dos extensos cabos de interconexão. Normalmente, em uma WAN, os hosts e a sub-rede são proprietários e administrados por diferentes pessoas. Em nosso exemplo, os funcionários poderiam ser responsáveis por seus próprios computadores, enquanto o departamento de TI da empresa está encarregado do restante da rede. Veremos limites mais claros nos próximos exemplos, em que o provedor da rede ou a companhia telefônica opera a sub-rede. A separação dos aspectos de comunicação estritos da rede (a sub-rede) dos aspectos da aplicação (os hosts) simplifica bastante o projeto geral da rede.

Uma segunda diferença é que os roteadores normalmente conectarão diferentes tipos de tecnologia de rede. As redes dentro dos escritórios podem ser Ethernet comutada, por exemplo, enquanto as linhas de transmissão de longa distância podem ser enlaces SONET (que veremos no Capítulo 2). Algum dispositivo é necessário para juntá-las. O leitor atento notará que isso vai além da nossa definição de uma rede. Isso significa que muitas WANs de fato serão **redes interligadas**, ou redes compostas que são criadas a partir de mais de uma rede. Voltaremos a esse assunto sobre redes interligadas na próxima seção.

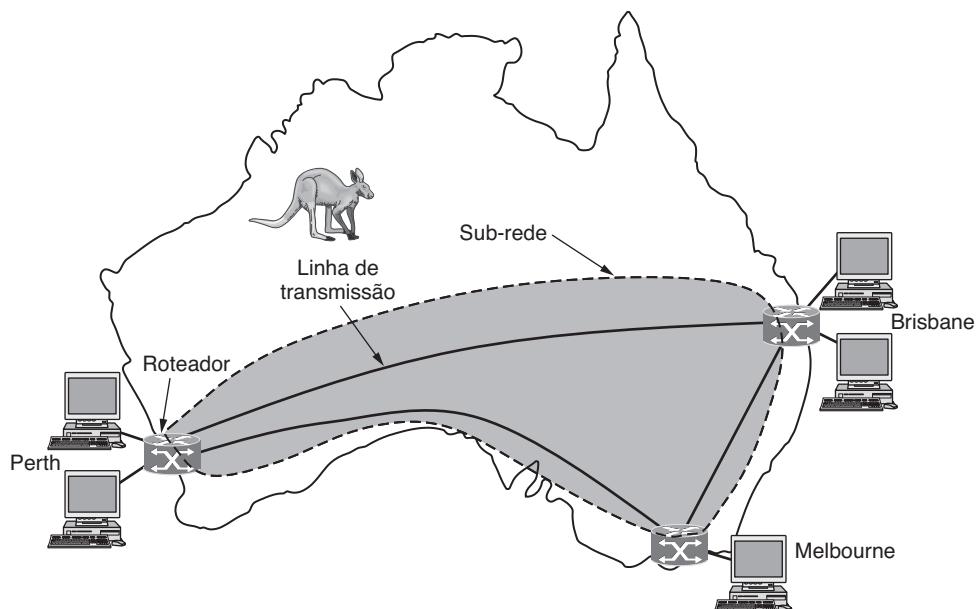



Figura 1.8 | WAN que conecta três escritórios de filiais na Austrália.

Uma última diferença é naquilo que é conectado à sub-rede. Podem ser computadores individuais, como foi o caso para a conexão às LANs, ou podem ser LANs inteiras. É assim que redes maiores são montadas a partir de redes menores. Em relação à sub-rede, ela tem a mesma função.

Agora estamos em uma posição de examinar duas outras variedades de WANs. Primeiro, em vez de alugar linhas de transmissão dedicadas, uma empresa pode conectar seus escritórios à Internet. Isso permite que as conexões sejam feitas entre os escritórios como enlaces virtuais que usam a capacidade de infraestrutura da Internet. Esse arranjo, mostrado na Figura 1.9, é chamado de **rede privada virtual**, ou **VPN** (**Virtual Private Network**). Em comparação com o arranjo dedicado, uma VPN tem a vantagem comum da virtualização, ou seja, ela oferece flexibilidade na reutilização de recurso (conectividade com a Internet). Para entender isso, considere como é fácil incluir um quarto escritório. Uma VPN também tem a desvantagem normal da virtualização, o que é uma falta de controle sobre os recursos subjacentes. Com uma linha dedicada, a capacidade é clara. Com uma VPN, suas milhas estão sujeitas à variação, de acordo com o serviço da Internet.

A segunda variação é que a sub-rede pode ser operada por uma empresa diferente. O operador da sub-rede é conhecido como um **provedor de serviço de rede**, e os escritórios são seus clientes. Essa estrutura aparece na Figura 1.10. O operador da sub-rede também se conectará a outros clientes, desde que eles possam pagar e ela possa oferecer o serviço. Como seria decepcionante um serviço de rede em que os clientes só pudessem enviar pacotes uns aos outros, o operador da sub-rede também se conectará a outras redes que fazem parte da Internet. Esse operador de sub-rede é chamado de **provedor de serviço de Internet**, ou **ISP** (**Internet Service Provider**), e a sub-rede é uma **rede ISP**. Seus clientes que se conectam à ISP recebem serviço de Internet.

Podemos usar a rede ISP para prever algumas questões fundamentais que estudaremos em outros capítulos. Na maioria das WANs, a rede contém muitas linhas de transmissão, cada uma conectando um par de roteadores. Se dois roteadores que não compartilham uma linha de transmissão quiserem se comunicar, eles precisam fazer isso indiretamente, por meio de outros roteadores. Pode haver muitos caminhos na rede que conectam esses dois roteadores. O processo em que o roteador toma a decisão sobre qual caminho usar e para onde enviar o pacote em seguida, para a interface adequada, é chamado de **algoritmo de roteamento**. Existem muitos desses algoritmos. Estudaremos alguns tipos em detalhes no Capítulo 5.

Outros tipos de WANs utilizam muito as tecnologias sem fio. Nos sistemas via satélite, cada computador no solo tem uma antena através da qual ele pode enviar e receber dados de e para um satélite em órbita. Todos os computadores podem escutar a saída *do* satélite, e em alguns casos eles também podem escutar as transmissões que sobem de seus computadores *para* o satélite da mesma forma. As redes de satélite são inerentemente de radiodifusão, e são mais úteis quando essa propriedade é importante.

A rede de telefonia celular é outro exemplo de uma WAN que usa tecnologia sem fio. Esse sistema já passou por três gerações, e uma quarta está a caminho. A primeira geração era analógica e usada apenas para voz. A segunda geração era digital e apenas para voz. A terceira geração é digital e se destina a voz e dados. Cada estação-base celular cobre uma distância muito maior do que uma LAN sem fio, com um alcance medido em quilômetros, ao invés de dezenas de

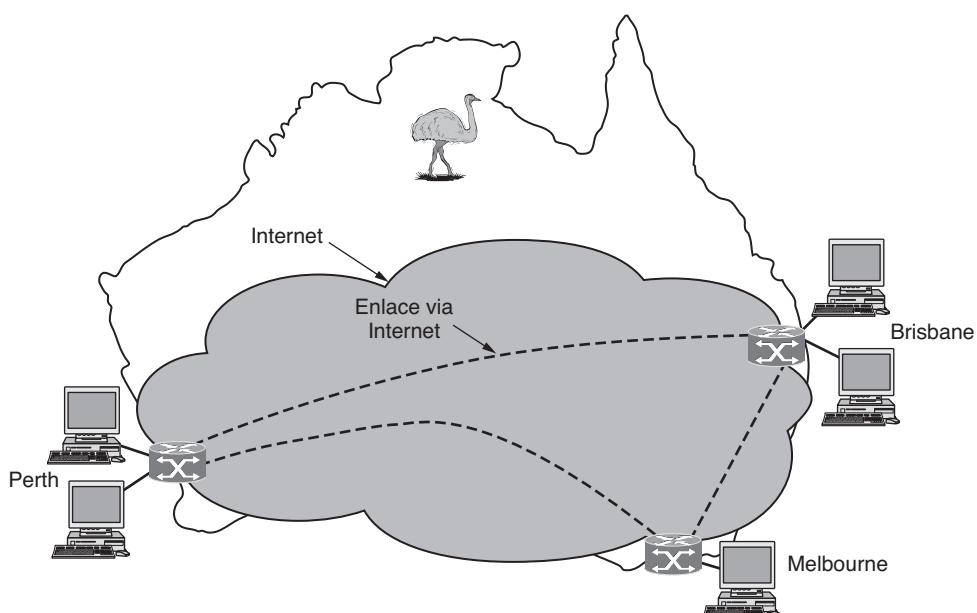



Figura 1.9 | WAN usando uma rede privada virtual.

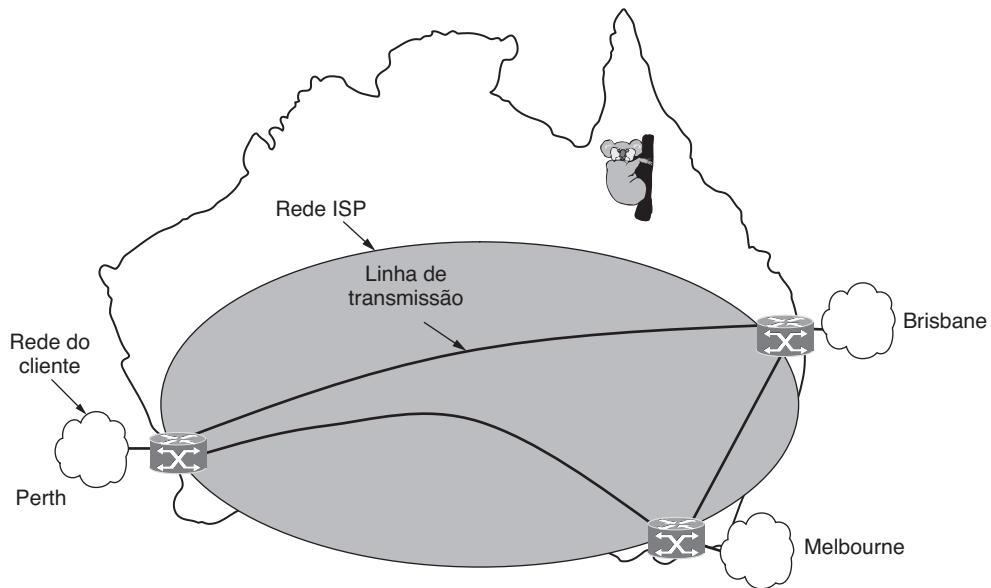



Figura 1.10 | WAN usando uma rede ISP.

metros. As estações-base são conectadas umas às outras por uma rede de backbone que normalmente é conectada por cabos. As taxas de dados das redes celulares normalmente estão na ordem de 1 Mbps, muito menos do que uma LAN sem fio, que pode chegar a uma ordem de 100 Mbps. Falaremos bastante sobre essas redes no Capítulo 2.

1.2.5 Redes interligadas (internets)

Existem muitas redes no mundo, frequentemente apresentando diferentes tipos de hardware e software. Normalmente, as pessoas conectadas a redes distintas precisam se comunicar entre si. Para que esse desejo se torne uma realidade, é preciso que se estabeleçam conexões entre redes quase sempre incompatíveis. Um conjunto de redes interconectadas forma uma **rede interligada** ou **internet**. Esses termos serão usados em um sentido genérico, em contraste com a Internet mundial (uma rede interligada em nível mundial), que sempre será representada com inicial maiúscula. A Internet usa redes ISP para conectar redes empresariais, domésticas e muitas outras redes. Veremos a Internet com muito mais detalhes em outro ponto deste livro.

Em geral, sub-redes, redes e redes interligadas se confundem. Uma sub-rede faz mais sentido no contexto de uma rede a longa distância, em que ela se refere ao conjunto de roteadores e linhas de comunicação pertencentes à operadora da rede. Como analogia, o sistema telefônico consiste em estações de comutação telefônica conectadas entre si por linhas de alta velocidade, e com casas e escritórios por linhas de baixa velocidade. Essas linhas e equipamentos, cuja propriedade e gerenciamento são da empresa de telefonia, formam a sub-rede do sistema telefônico. Os telefones propriamente ditos (os hosts, nessa analogia) não fazem parte da sub-rede.

Uma rede interligada é formada pela combinação de uma sub-rede e seus hosts. Entretanto, a palavra 'rede' é normalmente usada também em um sentido mais livre. Uma sub-rede poderia ser descrita como uma rede, como no caso da 'rede ISP' da Figura 1.10. Uma rede interligada também pode ser descrita como uma rede, como no caso da WAN na Figura 1.8. Seguiremos uma prática semelhante e, se estivermos distinguindo uma rede de outros arranjos, ficaremos com nossa definição original de uma coleção de computadores interconectados por uma única tecnologia.

Falaremos mais sobre o que constitui uma rede interligada. Sabemos que ela é formada quando redes distintas são interconectadas. Em nossa visão, a conexão entre uma LAN e uma WAN ou a conexão de duas LANs é o modo normal de formar uma rede interligada, mas existe pouco acordo no setor sobre a terminologia nessa área. Existem duas regras práticas que são úteis. Primeiro, se diferentes organizações pagam pela construção de partes distintas da rede e cada uma mantém sua parte, temos uma rede interligada, e não uma única rede. Segundo, se a tecnologia subjacente é diferente em partes distintas (por exemplo, broadcast *versus* ponto a ponto e cabeada *versus* sem fio), provavelmente temos uma rede interligada.

Indo mais a fundo, precisamos falar sobre como duas redes diferentes podem ser conectadas. O nome geral para uma máquina que faz uma conexão entre duas ou mais redes e oferece a conversão necessária, tanto em termos de hardware quanto de software, é um **gateway**. Os gateways são distinguidos pela camada em que operam na hierarquia de protocolos. Falaremos mais sobre camadas e hierarquias de protocolos a partir da próxima seção, mas, por enquanto, imagine que as camadas mais altas são mais ligadas às

aplicações, como a Web, e as camadas mais baixas são mais ligadas a enlaces de transmissão, como a Ethernet.

Como o benefício de formar uma rede interligada é conectar computadores pelas redes, não queremos usar um gateway em muito baixo nível, ou então não poderemos fazer conexões entre diferentes tipos de redes. Também não queremos usar um gateway em um nível muito alto, ou então a conexão só funcionará para determinadas aplicações. O nível do meio, que é o mais apropriado, normalmente é chamado de camada de rede, e um roteador é um gateway que comuta pacotes nessa camada. Agora podemos localizar uma rede interligada descobrindo uma rede que tem roteadores.

1.3 Software de rede

No projeto das primeiras redes de computadores, o hardware foi a principal preocupação e o software ficou em segundo plano. Essa estratégia foi deixada para trás. Atualmente, o software de rede é altamente estruturado. Nas próximas seções, examinaremos com alguns detalhes a técnica de estruturação do software. O método descrito aqui é de fundamental importância para o livro inteiro e faremos repetidas referências a ele.

1.3.1 Hierarquias de protocolos

Para reduzir a complexidade de seu projeto, a maioria das redes é organizada como uma pilha de **camadas** (ou **níveis**), colocadas umas sobre as outras. O número, o nome, o conteúdo e a função de cada camada diferem de uma rede para outra. No entanto, em todas as redes o objetivo de cada camada é oferecer determinados serviços às camadas superiores, isolando essas camadas dos detalhes de implementação real desses recursos. Em certo sentido, cada camada é uma espécie de máquina virtual, oferecendo determinados serviços à camada situada acima dela.

Na realidade, esse conceito é familiar e utilizado em toda a ciência da computação, na qual é conhecido por nomes diferentes, como ocultação de informações, tipos de dados abstratos, encapsulamento de dados e programação orientada a objetos. A ideia fundamental é que um determinado item de software (ou hardware) forneça um serviço a seus usuários, mas mantenha ocultos os detalhes de seu estado interno e de seus algoritmos.

Quando a camada *n* de uma máquina se comunica com a camada *n* de outra máquina, coletivamente, as regras e convenções usadas nesse diálogo são conhecidas como o protocolo da camada *n*. Basicamente, um **protocolo** é um acordo entre as partes que se comunicam, estabelecendo como se dará a comunicação. Como analogia, quando uma mulher é apresentada a um homem, ela pode estender a mão para ele que, por sua vez, pode apertá-la ou beijá-la, dependendo, por exemplo, do fato de ela ser uma advogada norte-americana que esteja participando de uma reunião de negócios ou uma princesa europeia presente em um baile de gala. A violação do protocolo dificultará a comunicação, se não torná-la completamente impossível.

A Figura 1.11 ilustra uma rede de cinco camadas. As entidades que ocupam as camadas correspondentes em diferentes máquinas são chamadas pares (ou **peers**). Os pares podem ser processos de software, dispositivos de hardware, ou mesmo seres humanos. Em outras palavras, são os pares que se comunicam utilizando o protocolo.

Na realidade, os dados não são transferidos diretamente da camada *n* de uma máquina para a camada *n* em outra máquina. Em vez disso, cada camada transfere os dados e as informações de controle para a camada imediatamente abaixo dela, até a camada mais baixa ser alcançada. Abaixo da camada 1 encontra-se o **meio físico** por meio do qual se dá a comunicação propriamente dita. Na Figura 1.11, a comunicação virtual é mostrada por linhas pontilhadas e a comunicação física, por linhas contínuas.

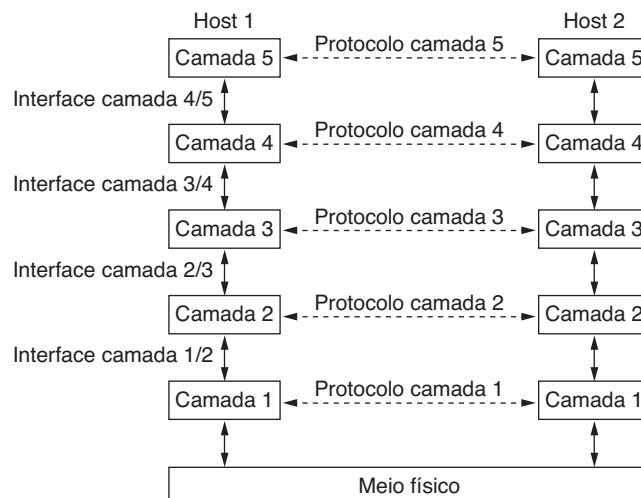



Figura 1.11 | Camadas, protocolos e interfaces.

Entre cada par de camadas adjacentes existe uma **interface**. Esta define as operações e os serviços que a camada inferior tem a oferecer à camada que se encontra acima dela. Quando os projetistas de rede decidem a quantidade de camadas que será incluída em uma rede e o que cada uma delas deve fazer, uma das considerações mais importantes é a definição de interfaces claras entre as camadas. Por sua vez, isso exige que cada camada execute um conjunto específico de funções bem definidas. Além de reduzir o volume de informações que deve ser passado de uma camada para outra, as interfaces bem definidas também simplificam a substituição de uma camada por um protocolo ou implementação completamente diferente (por exemplo, a substituição de todas as linhas telefônicas por canais de satélite), pois o novo protocolo ou a nova implementação só precisa oferecer exatamente o mesmo conjunto de serviços à sua vizinha de cima, assim como era feito na implementação anterior. De fato, é comum que hosts diferentes utilizem implementações distintas do mesmo protocolo (normalmente, escrito por empresas diferentes). De fato, o próprio protocolo pode mudar em alguma camada sem que as camadas acima e abaixo dela sequer percebam.

Um conjunto de camadas e protocolos é chamado **arquitetura de rede**. A especificação de uma arquitetura deve conter informações suficientes para permitir que um implementador desenvolva o programa ou construa o hardware de cada camada de forma que ela obedeça corretamente ao protocolo adequado. Nem os detalhes da implementação nem a especificação das interfaces pertencem à arquitetura, pois tudo fica oculto dentro das máquinas e não é visível do exterior. Nem sequer é necessário que as interfaces de todas as máquinas de uma rede sejam iguais, desde que cada uma delas possa usar todos os protocolos da maneira correta. Uma lista de protocolos usados por um determinado sistema, um protocolo por camada, é chamada **pilha de protocolos**. Arquiteturas de rede, pilhas de protocolos e os próprios protocolos são os principais assuntos deste livro.

Uma analogia pode ajudar a explicar a ideia de uma comunicação em várias camadas. Imagine dois filósofos (processos pares na camada 3), um dos quais fala urdu e português e o outro fala chinês e francês. Como não falam um idioma comum, eles contratam tradutores (processos pares na camada 2), que por sua vez têm cada qual uma secretária (processos pares na camada 1). O filósofo 1 deseja transmitir sua predileção por *oryctolagus cuniculus* a seu par. Para tal, ele envia uma mensagem (em português) através da interface 2/3 a seu tradutor, na qual diz 'Gosto de coelhos', como mostra a Figura 1.12. Como os tradutores resolveram usar um idioma neutro, o holandês, a mensagem foi convertida para 'Ik vind konijnen leuk'. A escolha do idioma é o protocolo da camada 2, que deve ser processada pelos pares dessa camada.

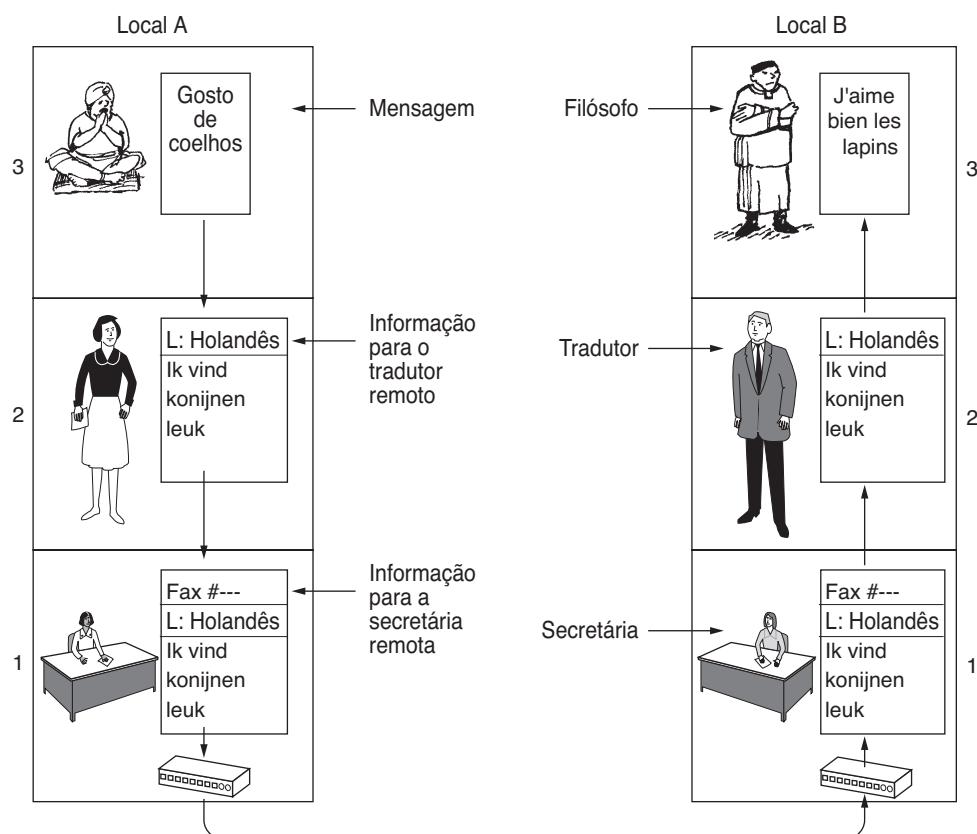



Figura 1.12 | A arquitetura filósofo-tradutor-secretária.

O tradutor entrega a mensagem a uma secretária para ser transmitida, por exemplo, por fax (o protocolo da camada 1). Quando chega, a mensagem é traduzida para o francês e passada através da interface 2/3 para o filósofo 2. Observe que cada protocolo é totalmente independente dos demais, desde que as interfaces não sejam alteradas. Nada impede que os tradutores mudem do holandês para o finlandês, desde que ambos concordem com a modificação e que ela não afete sua interface com a camada 1 ou com a camada 3. De modo semelhante, as secretárias podem passar de fax para correio eletrônico ou telefone sem incomodar (ou mesmo informar) as outras camadas. Cada processo só pode acrescentar informações dirigidas a seu par. Essas informações não são enviadas à camada superior.

Vejamos agora um exemplo mais técnico: como oferecer comunicação à camada superior da rede de cinco camadas na Figura 1.13. Uma mensagem, *M*, é produzida por um processo de aplicação que funciona na camada 5 e é entregue à camada 4 para transmissão. A camada 4 coloca um **cabeçalho** no início da mensagem para identificá-la e envia o resultado à camada 3. O cabeçalho inclui informações de controle, como endereços, a fim de permitir que a camada 4 da máquina de destino entregue a mensagem. Outros exemplos de informação de controle usados em algumas camadas são números de sequência (caso a camada inferior não preserve a ordem da mensagem), tamanhos e tempos.

Em muitas redes, não há limite para o tamanho das mensagens transmitidas no protocolo da camada 4, mas quase sempre há um limite imposto pelo protocolo da camada 3. Consequentemente, a camada 3 deve dividir as mensagens recebidas em unidades menores, pacotes, anexando um cabeçalho da camada 3 a cada pacote. Nesse exemplo, *M* é dividido em duas partes, M_1 e M_2, que serão transmitidas separadamente.

A camada 3 decide as linhas de saída que serão usadas e transmite os pacotes à camada 2. Esta acrescenta não apenas um cabeçalho a cada fragmento, mas também um final, e fornece a unidade resultante à camada 1 para transmissão física. Na máquina receptora, a mensagem se move de baixo para cima, de camada a camada, com os cabeçalhos sendo retirados durante o processo. Nenhum dos cabeçalhos das camadas abaixo de *n* é repassado à camada *n*.

Para entender a Figura 1.13, é importante observar a relação entre a comunicação virtual e a comunicação real, e a diferença entre protocolos e interfaces. Por exemplo, para os processos pares na camada 4, sua comunicação é 'horizontal', utilizando o protocolo da camada 4. O procedimento de cada um deles tem um nome semelhante a *EnviarParaOutroLado* e *ReceberDoOutroLado*, muito embora esses procedimentos na realidade se comuniquem com camadas inferiores através da interface 3/4, e não com o outro lado.

A abstração de processos pares (peers) é fundamental para toda a estrutura da rede. Com sua utilização, a tarefa não gerenciável de projetar a rede completa pode ser dividida em diversos problemas de projeto menores e gerenciáveis, ou seja, o projeto das camadas individuais.

Embora o título da Seção 1.3 seja 'Software de rede', vale a pena lembrar que as camadas inferiores de uma hierarquia de protocolos costumam ser implementadas no hardware ou como firmware. Apesar disso, algoritmos de protocolo muito complexos estão envolvidos no processo, mesmo se estiverem embutidos (parcial ou totalmente) no hardware.

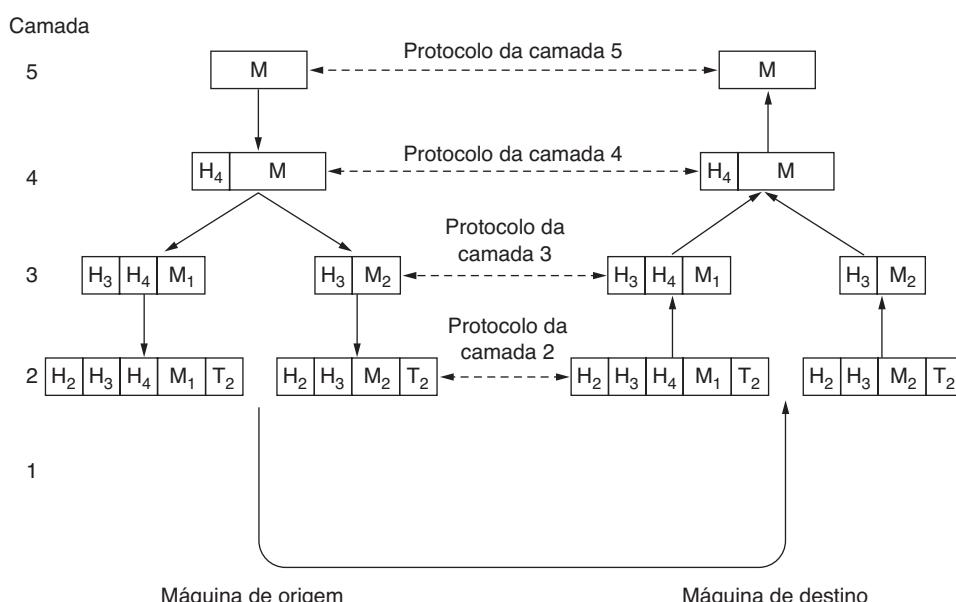



Figura 1.13 | Exemplo de fluxo de informações que admite a comunicação virtual na camada 5.

1.3.2 Questões de projeto relacionadas às camadas

Algumas questões fundamentais de projeto que ocorrem em redes de computadores estarão presentes em diversas camadas. Mencionaremos a seguir algumas das questões mais importantes.

Confiabilidade é a questão de projeto de criar uma rede que opere corretamente, embora sendo composta de uma coleção de componentes que não são confiáveis. Pense nos bits de um pacote trafegando pela rede. Há uma chance de que alguns desses bits sejam recebidos com problemas (invertidos) em virtude de um ruído elétrico casual, sinais sem fio aleatórios, falhas de hardware, bugs de software e assim por diante. Como é possível encontrar e consertar esses erros?

Um mecanismo para localizar erros na informação recebida usa códigos para **detecção de erros**. As informações recebidas incorretamente podem, então, ser retransmitidas até que sejam recebidas corretamente. Códigos mais poderosos permitem a **correção de erro**, em que a mensagem correta é recuperada a partir de bits possivelmente incorretos, que foram recebidos originalmente. Esses dois mecanismos funcionam acrescentando informações redundantes. Eles são usados nas camadas baixas, para proteger os pacotes enviados por enlaces individuais, e nas camadas altas, para verificar se o conteúdo correto foi recebido.

Outra questão de confiabilidade é descobrir um caminho que funcione através de uma rede. Normalmente, existem vários caminhos entre origem e destino e, em uma rede grande, pode haver alguns enlaces ou roteadores com defeito. Suponha que a rede esteja parada na Alemanha. Os pacotes enviados de Londres a Roma pela Alemanha não passarão, mas poderíamos enviar pacotes de Londres para Roma via Paris. A rede deve tomar essa decisão automaticamente. Esse tópico é chamado **roteamento**.

Uma segunda questão de projeto se refere à evolução da rede. Com o tempo, as redes se tornam maiores e novos projetos aparecem precisando ser conectados à rede existente. Recentemente, vimos o mecanismo-chave de estrutura usado para dar suporte à mudança, dividindo o problema geral e ocultando detalhes da implementação: as **camadas de protocolos**. Mas existem muitas outras estratégias.

Como existem muitos computadores na rede, cada camada precisa de um mecanismo para identificar transmissores e receptores que estão envolvidos em uma determinada mensagem. Esse mecanismo é conhecido como **endereçamento** ou **nomeação**, nas camadas alta e baixa, respectivamente.

Um aspecto do crescimento é que diferentes tecnologias de rede normalmente têm diferentes limitações. Por exemplo, nem todos os canais de comunicação preservam a ordem das mensagens enviadas neles, ocasionando soluções que numeram mensagens. Outro exemplo são as diferenças no tamanho máximo de uma mensagem que as redes podem transmitir. Isso ocasiona mecanismos para dividir, transmitir e depois juntar novamente as mensagens. Esse tópico geral é chamado de **interligação de redes**.

Quando as redes ficam muito grandes, novos problemas aparecem. As cidades podem ter engarrafamentos no trânsito, falta de números de telefone, e é fácil se perder pelas ruas. Muitas pessoas não têm esses problemas em sua própria vizinhança, mas a cidade inteira pode ser um grande problema. Os projetos que continuam a crescer bem enquanto a rede cresce são considerados **escaláveis**.

Uma terceira questão de projeto é a alocação de recursos. As redes oferecem um serviço aos hosts a partir de seus recursos subjacentes, como a capacidade de linhas de transmissão. Para fazer isso bem, elas precisam de mecanismos que dividem seus recursos de modo que um host não interfira muito em outro.

Muitos projetos compartilham a largura de banda da rede dinamicamente, de acordo com a necessidade dos hosts a curto prazo, ao invés de dar a cada host uma fração fixa da largura de banda, que ele pode ou não usar. Esse projeto é chamado **multiplexação estatística**, significando compartilhar com base nas estatísticas de demanda. Ele pode ser aplicado às camadas inferiores para um único enlace, nas camadas altas para uma rede ou mesmo em aplicações que usam a rede.

Uma questão de alocação que afeta cada nível é como impedir que um transmissor rápido envie uma quantidade excessiva de dados a um receptor mais lento. Normalmente, usa-se uma espécie de *feedback* do receptor para o transmissor. Esse tópico é chamado **controle de fluxo**. Às vezes, o problema é que a rede é sobrecarregada porque muitos computadores querem enviar muito tráfego e a rede não pode entregar tudo isso. A sobrecarga da rede é chamada **congestionamento**. Uma estratégia é que cada computador reduza sua demanda quando experimentar um congestionamento. Isso também pode ser usado em todas as camadas.

É interessante observar que a rede tem mais recursos a oferecer do que simplesmente largura de banda. Para usos como o transporte de vídeo ao vivo, a prontidão na entrega importa muito. A maioria das redes precisa oferecer serviço às aplicações que desejam essa entrega em **tempo real** ao mesmo tempo que oferece serviço a aplicações que desejam uma alta vazão. A **qualidade de serviço** é o nome dado aos mecanismos que reconciliam essas demandas concorrentes.

A última questão de projeto trata de proteger a rede, defendendo-a contra diferentes tipos de ameaças. Uma das ameaças que mencionamos anteriormente é a de bisbilhotagem nas comunicações. Mecanismos que oferecem **confidencialidade** defendem contra essa ameaça e são usados em várias camadas. Os mecanismos para **autenticação** impedem que alguém finja ser outra pessoa. Eles poderiam

ser usados para diferenciar websites falsos de um banco real, ou para permitir verificar se uma chamada da rede celular está realmente vindo de seu telefone, para que você pague a conta correta. Outros mecanismos para **integridade** impedem mudanças clandestinas nas mensagens, como alterar 'debite US$ 10 da minha conta' para 'debite US$ 1.000 da minha conta'. Todos esses projetos são baseados em criptografia, que estudaremos no Capítulo 8.

1.3.3 Serviços orientados e não orientados a conexões

As camadas podem oferecer dois tipos diferentes de serviços às camadas situadas acima delas: serviços orientados a conexões e serviços não orientados a conexões. Nesta seção, examinaremos esses dois tipos de serviços e as diferenças entre eles.

O serviço **orientado a conexões** se baseia no sistema telefônico. Para falar com alguém, você tira o fone do gancho, digita o número, fala e, em seguida, desliga. Da mesma forma, para utilizar um serviço de rede orientado a conexões, primeiro o usuário do serviço estabelece uma conexão, a utiliza, e depois a libera. O aspecto essencial de uma conexão é que ela funciona como um tubo: o transmissor empurra objetos (bits) em uma extremidade, e esses objetos são recebidos pelo receptor na outra extremidade. Na maioria dos casos, a ordem é preservada, de forma que os bits chegam na sequência em que foram enviados.

Em alguns casos, quando uma conexão é estabelecida, o transmissor, o receptor e a sub-rede conduzem uma **negociação** sobre os parâmetros a serem usados, como o tamanho máximo das mensagens, a qualidade do serviço exigida e outras questões. Em geral, um lado faz uma proposta e a outra parte pode aceitá-la, rejeitá-la ou fazer uma contraproposta. Um **circuito** é outro nome para uma conexão com recursos associados, como uma largura de banda fixa. Isso vem desde a rede telefônica, em que um circuito era um caminho pelo fio de cobre que transportava uma conversa telefônica.

Ao contrário do serviço orientado a conexões, o serviço **não orientado a conexões** se baseia no sistema postal. Cada mensagem (carta) carrega o endereço de destino completo e cada uma delas é roteada pelos nós intermediários através do sistema, independentemente de todas as outras. Existem diferentes nomes para mensagens em diferentes contextos; um **pacote** é uma mensagem na camada de rede. Quando os nós intermediários recebem uma mensagem completa antes de enviá-la para o próximo nó, isso é chamado **comutação store-and-forward**. A alternativa, em que a transmissão de uma mensagem em um nó começa antes de ser completamente recebida por ele, é chamada **comutação cut-through**. Normalmente, quando duas mensagens são enviadas ao mesmo destino, a primeira a ser enviada é a primeira a chegar. No entanto, é possível que a primeira mensagem a ser enviada esteja atrasada, de modo que a segunda chegue primeiro.

Cada tipo de serviço pode ser caracterizado por sua confiabilidade. Alguns serviços são confiáveis, no sentido de nunca perderem dados. Em geral, um serviço confiável é implementado para que o receptor confirme o recebimento de cada mensagem, de modo que o transmissor se certifique de que ela chegou. O processo de confirmação introduz overhead e atrasos, que normalmente compensam, mas às vezes são indesejáveis.

Uma situação típica em que um serviço orientado a conexões confiável é apropriado é a transferência de arquivos. O proprietário do arquivo deseja se certificar de que todos os bits chegaram corretamente e na mesma ordem em que foram enviados. São poucos os clientes de transferência de arquivos que preferem um serviço que ocasionalmente desorganiza ou perde alguns bits, mesmo que ele seja muito mais rápido.

O serviço orientado a conexões confiável tem duas variações secundárias: sequências de mensagens e fluxos de bytes. Na primeira variação, os limites das mensagens são preservados. Quando duas mensagens de 1.024 bytes são enviadas, elas chegam como duas mensagens distintas de 1.024 bytes, nunca como uma única mensagem de 2.048 bytes. Na segunda, a conexão é simplesmente um fluxo de bytes, sem limites de mensagem. Quando 2.048 bytes chegam ao receptor, não há como saber se eles foram enviados como uma mensagem de 2.048 bytes, duas mensagens de 1.024 bytes ou 2.048 mensagens de 1 byte. Se as páginas de um livro são enviadas por uma rede a uma fotocompositora como mensagens separadas, talvez seja importante preservar os limites da mensagem. Por outro lado, para baixar um filme de DVD, um fluxo de bytes do servidor para o computador do usuário é tudo o que é necessário. Os limites de mensagens dentro do filme não são relevantes.

Para algumas aplicações, os atrasos introduzidos pelas confirmações são inaceitáveis. Uma dessas aplicações é o tráfego de voz digital por **Voice over IP** (**VoIP**). Os usuários de telefone preferem ouvir um pouco de ruído na linha ou uma palavra truncada de vez em quando a experimentar um atraso para aguardar confirmações. O mesmo acontece durante a transmissão de uma conferência de vídeo; não haverá problema se aparecerem alguns pixels errados. No entanto, é irritante ver uma imagem parada enquanto o fluxo é interrompido e reiniciado para a correção de erros.

Nem todas as aplicações precisam do serviço orientado a conexões. Por exemplo, spammers enviam lixo eletrônico do serviço orientado a muitos destinatários. Provavelmente, o spammer não deseja enfrentar o problema de configurar e depois desfazer uma conexão apenas para enviar um item. Além disso, não será essencial uma entrega 100 por cento confiável, em especial se o custo for maior. É necessário apenas um modo de enviar uma única mensagem que tenha uma alta probabilidade de chegar, mas sem garantias. O ser-

viço não orientado a conexões não confiável (ou seja, sem confirmação) costuma ser chamado serviço de **datagramas**, em uma analogia com o serviço de telegramas, que também não oferece uma confirmação ao transmissor. Apesar de não ser confiável, essa é a forma dominante na maioria das redes, por motivos que adiante se tornarão mais claros.

Em outras situações, a conveniência de não ter de estabelecer uma conexão para enviar uma única mensagem curta é desejável, mas a confiabilidade é essencial. O serviço de **datagramas confirmados** pode ser oferecido para essas aplicações. Ele é semelhante a enviar uma carta registrada e solicitar um aviso de recebimento. Quando o aviso é devolvido, o transmissor fica absolutamente certo de que a carta foi entregue ao destinatário e não perdida ao longo do caminho. As mensagens de texto em telefones móveis são um exemplo.

Outro serviço é o de **solicitação/resposta**. Nele, o transmissor envia um único datagrama contendo uma solicitação; a resposta contém a réplica. A solicitação/resposta é em geral usada para implementar a comunicação no modelo cliente-servidor: o cliente emite uma solicitação e o servidor responde. Por exemplo, um cliente de telefone móvel poderia enviar uma consulta a um servidor de mapa para receber os dados de mapa para seu local atual. A Figura 1.14 resume os tipos de serviços descritos anteriormente.

O conceito de usar comunicação não confiável pode ser confuso a princípio. Afinal de contas, por que alguém iria preferir uma comunicação não confiável à comunicação confiável? Em primeiro lugar, a comunicação confiável (em nosso sentido, isto é, confirmada) pode não estar disponível em uma determinada camada. Por exemplo, a Ethernet não fornece comunicação confiável. Ocasionalmente, os pacotes podem ser danificados em trânsito. Cabe aos níveis mais altos do protocolo lidar com esse problema. Em particular, muitos serviços confiáveis são montados em cima de um serviço de datagrama não confiável. Em segundo lugar, os atrasos inerentes ao fornecimento de um serviço confiável podem ser inaceitáveis, em especial nas aplicações em tempo real, como as de multimídia. Por essas razões, coexistem tanto a comunicação confiável quanto a não confiável.

1.3.4 Primitivas de serviço

Um serviço é especificado formalmente por um conjunto de **primitivas** (operações) disponíveis para que os processos do usuário acessem o serviço. Essas primitivas informam ao serviço que ele deve executar alguma ação ou relatar uma ação executada por uma entidade par. Se a pilha de protocolos estiver localizada no sistema operacional, como ocorre com frequência, as primitivas serão normalmente chamadas do sistema. Essas chamadas geram uma armadilha para o kernel, que então devolve o controle da máquina ao sistema operacional para enviar os pacotes necessários.

O conjunto de primitivas disponíveis depende da natureza do serviço que está sendo fornecido. As primitivas para um serviço orientado a conexões são diferentes das oferecidas em um serviço não orientado a conexões. Como um exemplo mínimo das primitivas de serviço que poderiam ser fornecidas para implementar um fluxo de bytes confiável, considere as primitivas listadas na Tabela 1.3. Elas serão familiares para os fãs do soquete de Berkeley, pois as primitivas são uma versão simplificada dessa interface.

Essas primitivas podem ser usadas para uma interação de solicitação/resposta em um ambiente cliente/servidor. Para ilustrar como, esboçamos um protocolo simples que implementa o serviço usando datagramas confirmados.

Primeiro, o servidor executa LISTEN para indicar que está preparado para aceitar conexões de entrada. Um caminho comum para implementar LISTEN é torná-la uma chamada de bloqueio do sistema. Depois de executar a primitiva, o processo servidor fica bloqueado até que surja uma solicitação de conexão.

Em seguida, o processo cliente executa CONNECT para estabelecer uma conexão com o servidor. A chamada CONNECT precisa especificar a quem se conectar; assim, ela poderia ter um parâmetro fornecendo o endereço do servidor. Então, em geral, o sistema operacional envia um pacote ao par solicitando que ele se conecte, como mostra o item (1) na Figura 1.15. O processo cliente é suspenso até haver uma resposta.

	Serviço	Exemplo
Orientados a conexões	Fluxo de mensagens confiável	Sequência de páginas
	Fluxo de bytes confiável	Download de filme
	Conexão não confiável	VoIP
Sem conexões	Datagrama não confiável	Lixo de correio eletrônico
	Datagrama confirmado	Mensagem de texto
	Solicitação/resposta	Consulta a banco de dados

Figura 1.14 | Seis diferentes tipos de serviço.

Primitiva	Significado
LISTEN	Bloco que espera por uma conexão de entrada
CONNECT	Estabelecer uma conexão com um par que está à espera
ACCEPT	Aceitar uma conexão de entrada de um par
RECEIVE	Bloco que espera por uma mensagem de entrada
SEND	Enviar uma mensagem ao par
DISCONNECT	Encerrar uma conexão

Tabela 1.3 | Seis primitivas de serviço para implementação de uma conexão simples.

Quando o pacote chega ao servidor, o sistema operacional vê que o pacote está solicitando uma conexão. Ele verifica se existe um ouvinte e, se houver, desbloqueia o ouvinte. O processo servidor pode, então, estabelecer uma conexão com a chamada ACCEPT. Isso envia de volta uma confirmação (2) ao processo cliente para aceitar a conexão. A chegada dessa resposta libera o cliente. Nesse momento, o cliente e o servidor estão em execução e têm uma conexão estabelecida entre eles.

A analogia óbvia entre esse protocolo e a vida real ocorre quando um consumidor (cliente) liga para o gerente do serviço de atendimento ao consumidor de uma empresa. No início do dia, o gerente de serviço inicia a sequência ficando próximo ao telefone para atendê-lo caso ele toque. Mais tarde, o cliente efetua a chamada. Quando o gerente levanta o fone do gancho, a conexão é estabelecida.

A próxima etapa é a execução de RECEIVE pelo servidor, a fim de se preparar para aceitar a primeira solicitação. Normalmente, o servidor faz isso imediatamente após ser liberado de LISTEN, antes de a confirmação poder retornar ao cliente. A chamada de RECEIVE bloqueia o servidor.

Depois, o cliente executa SEND para transmitir sua solicitação (3), seguida pela execução de RECEIVE para receber a resposta. A chegada do pacote de solicitação à máquina servidora desbloqueia o processo servidor, para que ele possa processar a solicitação. Depois de terminar o trabalho, ele utiliza SEND para enviar a resposta ao cliente (4). A chegada desse pacote desbloqueia o cliente, que agora pode examinar a resposta. Se tiver solicitações adicionais, o cliente poderá fazê-las nesse momento.

Ao terminar, ele utiliza DISCONNECT para encerrar a conexão (5). Em geral, uma DISCONNECT inicial é uma chamada de bloqueio, suspendendo o cliente e enviando um pacote ao servidor para informar que a conexão não é mais necessária. Quando o servidor recebe o pacote, ele próprio também emite uma DISCONNECT, confirmando o pacote do cliente e liberando a conexão (6). Quando o pacote do servidor retorna à máquina cliente, o processo cliente é liberado e a conexão é interrompida. Em resumo, é assim que funciona a comunicação orientada a conexões.

É claro que a vida não é tão simples assim. Muitos detalhes podem dar errado. O sincronismo pode estar incorreto (por exemplo, CONNECT ser executada antes de LISTEN), os pacotes podem ser perdidos e muito mais. Examinaremos todas essas questões com muitos detalhes mais adiante; porém, por enquanto, a Figura 1.15 resume o funcionamento possível de uma comunicação cliente-servidor com datagramas confirmados, de modo que podemos ignorar os pacotes perdidos.

Considerando-se que são necessários seis pacotes para completar esse protocolo, alguém poderia perguntar por que não é utilizado um protocolo não orientado a conexões. A resposta é que, em um mundo perfeito, esse tipo de protocolo poderia ser usado e, nesse caso, seriam necessários apenas dois pacotes: um para a solicitação e outro para a resposta. Entretanto, em face de mensagens extensas em qualquer sentido (por exemplo, um arquivo com vários megabytes), erros de transmissão e pacotes perdidos, a situação muda. Se a resposta consistisse em centenas de pacotes, alguns dos quais pudessem se

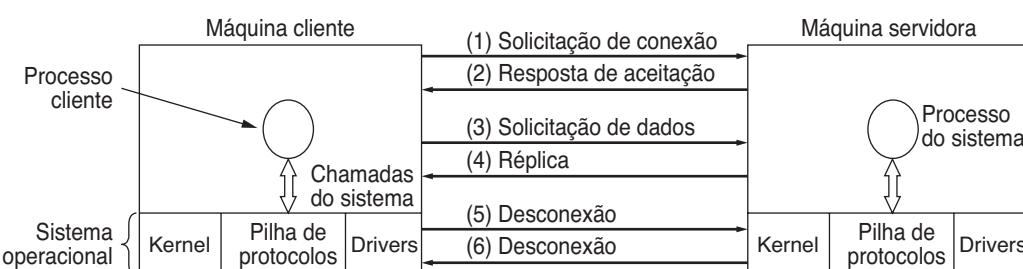



Figura 1.15 | Uma interação cliente-servidor simples, usando datagramas confirmados.

perder durante a transmissão, como o cliente saberia que alguns fragmentos se perderam? Como o cliente saberia que o último pacote recebido foi, de fato, o último pacote enviado? Suponha que o cliente quisesse um segundo arquivo. Como ele poderia distinguir o pacote 1 do segundo arquivo de um pacote 1 perdido do primeiro arquivo que repentinamente tivesse encontrado o caminho até o cliente? Em resumo, no mundo real, um simples protocolo de solicitação/resposta sobre uma rede não confiável normalmente é inadequado. No Capítulo 3, estudaremos em detalhes uma variedade de protocolos que superam esses e outros problemas. Por enquanto, basta dizer que às vezes ter um fluxo de bytes confiável e ordenado entre processos é muito conveniente.

1.3.5 Relacionamento entre serviços e protocolos

Serviços e protocolos são conceitos diferentes. Essa distinção é tão importante que vamos enfatizá-la mais uma vez. Um *serviço* é um conjunto de primitivas (operações) que uma camada oferece à camada situada acima dela. O serviço define as operações que a camada está preparada para executar em nome de seus usuários, mas não informa absolutamente nada sobre como essas operações são implementadas. Um serviço se relaciona a uma interface entre duas camadas, sendo a camada inferior o fornecedor do serviço e a camada superior, o usuário do serviço.

Ao contrário, o *protocolo* é um conjunto de regras que controla o formato e o significado dos pacotes ou mensagens que são trocadas pelas entidades pares contidas em uma camada. As entidades utilizam protocolos com a finalidade de implementar suas definições de serviço. Elas têm a liberdade de trocar seus protocolos, desde que não alterem o serviço visível para seus usuários. Portanto, o serviço e o protocolo são independentes um do outro. Esse é um conceito fundamental, que qualquer projetista de rede precisa entender bem.

Em outras palavras, os serviços estão relacionados às interfaces entre camadas, como ilustra a Figura 1.16. Por outro lado, os protocolos se relacionam aos pacotes enviados entre entidades pares em máquinas diferentes. É importante não confundir esses dois conceitos.

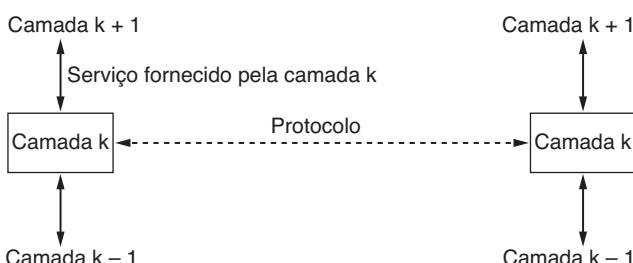



Figura 1.16 | Relacionamento entre um serviço e um protocolo.

Vale a pena fazer uma analogia com as linguagens de programação. Um serviço é como um objeto ou um tipo de dados abstrato em uma linguagem orientada a objetos. Ele define as operações que podem ser executadas sobre um objeto, mas não especifica como essas operações são implementadas. Em contraste, um protocolo se refere à *implementação* do serviço e, consequentemente, não é visível ao usuário do serviço.

Muitos protocolos mais antigos não distinguiam entre o serviço e o protocolo. Na prática, uma camada normal poderia ter uma primitiva de serviço send packet, com o usuário fornecendo um ponteiro para um pacote totalmente montado. Essa organização significava que todas as mudanças no protocolo ficavam imediatamente visíveis para os usuários. Hoje, a maioria dos projetistas de redes considera tal projeto um sério equívoco.

1.4 Modelos de referência

Depois de discutirmos o conceito de redes em camadas em termos abstratos, vamos ver alguns exemplos. Nas duas seções a seguir, examinaremos duas importantes arquiteturas de rede: os modelos de referência OSI e TCP/IP. Embora os *protocolos* associados ao modelo OSI raramente sejam usados nos dias de hoje, o *modelo* em si é de fato bastante geral e ainda válido, e as características descritas em cada camada ainda são muito importantes. O modelo TCP/IP tem características opostas: o modelo propriamente dito não é muito utilizado, mas os protocolos são bastante utilizados. Por essa razão, examinaremos ambos em detalhes. Além disso, às vezes é possível aprender mais com os erros do que com os acertos.

1.4.1 O modelo de referência OSI

O modelo OSI (exceto o meio físico) é representado na Figura 1.17. Esse modelo se baseia em uma proposta desenvolvida pela ISO (International Standards Organization) como um primeiro passo em direção à padronização internacional dos protocolos usados nas várias camadas (Day e Zimmermann, 1983). Ele foi revisado em 1995 (Day, 1995). O modelo se chama **Modelo de Referência ISO OSI** (**Open Systems Interconnection**), pois ele trata da interconexão de sistemas abertos — ou seja, sistemas abertos à comunicação com outros sistemas. Para abreviar, vamos chamá-lo simplesmente de **modelo OSI**.

O modelo OSI tem sete camadas. Veja, a seguir, um resumo dos princípios aplicados para chegar às sete camadas.

1. Uma camada deve ser criada onde houver necessidade de outro grau de abstração.
2. Cada camada deve executar uma função bem definida.

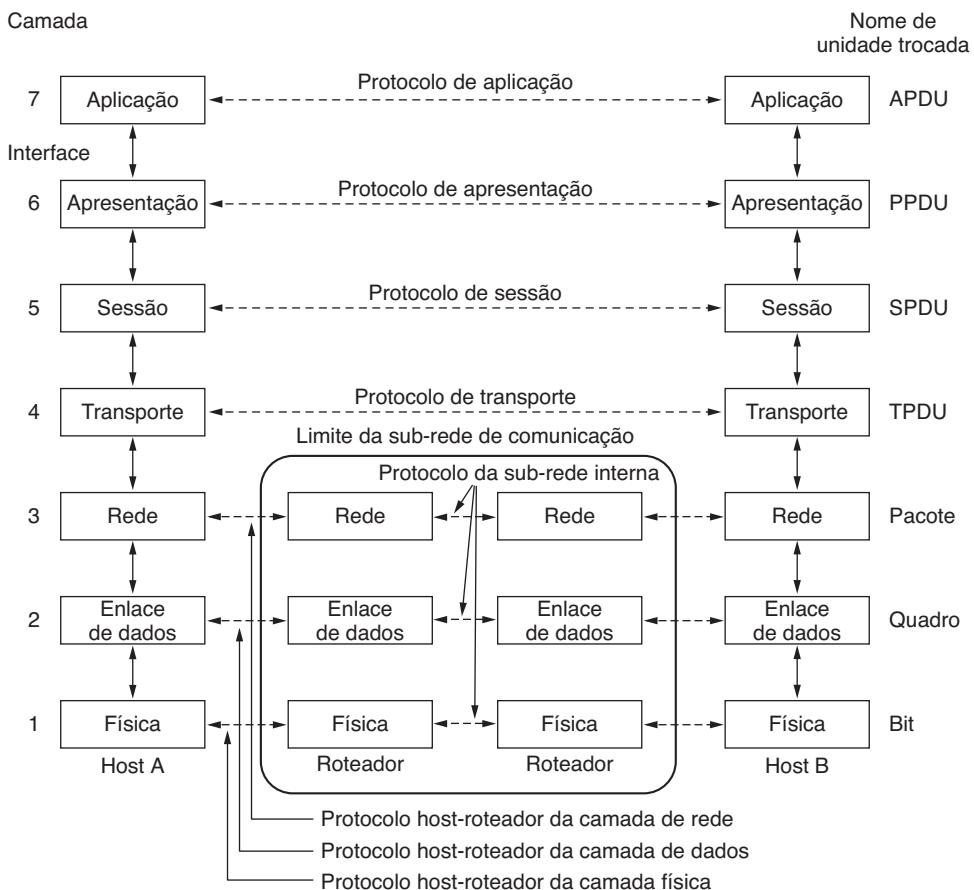



Figura 1.17 | O modelo de referência OSI.

3. A função de cada camada deve ser escolhida tendo em vista a definição de protocolos padronizados internacionalmente.
4. Os limites de camadas devem ser escolhidos para minimizar o fluxo de informações pelas interfaces.
5. O número de camadas deve ser grande o bastante para que funções distintas não precisem ser desnecessariamente colocadas na mesma camada e pequeno o suficiente para que a arquitetura não se torne difícil de controlar.

Em seguida, discutiremos cada uma das camadas do modelo, começando pela camada inferior. Observe que o modelo OSI propriamente dito não é uma arquitetura de rede, pois não especifica os serviços e protocolos exatos que devem ser usados em cada camada. Ele apenas informa o que cada camada deve fazer. No entanto, a ISO também produziu padrões para todas as camadas, embora esses padrões não façam parte do próprio modelo de referência. Cada um foi publicado como um padrão internacional distinto. O *modelo* (em parte) é bastante utilizado, embora os protocolos associados há muito tempo tenham sido deixados de lado.

A camada física

A **camada física** trata da transmissão de bits normais por um canal de comunicação. O projeto da rede deve garantir que, quando um lado enviar um bit 1, o outro lado o receberá como um bit 1, não como um bit 0. As questões mais comuns aqui são quais os sinais elétricos que devem ser usados para representar um bit 1 e um bit 0, a quantidade de nanossegundos que um bit deve durar, se a transmissão pode ou não ser realizada simultaneamente nos dois sentidos, a forma como a conexão inicial será estabelecida e de que maneira ela será encerrada quando ambos os lados tiverem terminado, e ainda quantos pinos o conector de rede terá e qual será a finalidade de cada pino. Nessa situação, as questões de projeto lidam em grande parte com interfaces mecânicas, elétricas e de sincronização, e com o meio físico de transmissão que se situa abaixo da camada física.

A camada de enlace de dados

A principal tarefa da **camada de enlace de dados** é transformar um canal de transmissão normal em uma linha que pareça livre de erros de transmissão. Para fazer isso, a camada de enlace mascara os erros reais, de modo

que a camada de rede não os veja. Isso é executado fazendo com que o transmissor divida os dados de entrada em **quadros de dados** (que, em geral, têm algumas centenas ou alguns milhares de bytes) e transmita os quadros sequencialmente. Se o serviço for confiável, o receptor confirmará a recepção correta de cada quadro, enviando de volta um **quadro de confirmação**.

Outra questão que surge na camada de enlace de dados (e na maioria das camadas mais altas) é como impedir que um transmissor rápido envie uma quantidade excessiva de dados a um receptor lento. Normalmente, é preciso que haja algum mecanismo que regule o tráfego para informar ao transmissor quando o receptor pode aceitar mais dados.

As redes de broadcast têm uma questão adicional a ser resolvida na camada de enlace de dados: como controlar o acesso ao canal compartilhado. Uma subcamada especial da camada de enlace de dados, a subcamada de **controle de acesso ao meio**, trata desse problema.

A camada de rede

A **camada de rede** controla a operação da sub-rede. Uma questão fundamental de projeto é determinar a maneira como os pacotes são roteados da origem até o destino. As rotas podem se basear em tabelas estáticas, 'amarradas' à rede e raramente alteradas, ou frequentemente podem ser atualizadas de forma automática, para evitar componentes defeituosos. Elas também podem ser determinadas no início de cada conversação; por exemplo, uma sessão de terminal, como um login em uma máquina remota. Por fim, elas podem ser altamente dinâmicas, sendo determinadas para cada pacote, refletindo a carga atual da rede.

Se houver muitos pacotes na sub-rede ao mesmo tempo, eles dividirão o mesmo caminho, formando gargalos. A responsabilidade pelo controle desse congestionamento também pertence à camada de rede, em conjunto com as camadas mais altas, que adaptam a carga imposta sobre a rede. De modo mais geral, a qualidade do serviço fornecido (atraso, tempo em trânsito, instabilidade etc.) também é uma questão da camada de rede.

Quando um pacote precisa trafegar de uma rede para outra até chegar a seu destino, podem surgir muitos problemas. O endereçamento utilizado pela segunda rede pode ser diferente do que é usado pela primeira. Talvez a segunda rede não aceite o pacote por ele ser muito grande. Os protocolos podem ser diferentes e assim por diante. Cabe à camada de rede superar todos esses problemas, a fim de permitir que redes heterogêneas sejam interconectadas.

Nas redes de broadcast, o problema de roteamento é simples e, assim, a camada de rede geralmente é estreita, ou mesmo inexistente.

A camada de transporte

A função básica da camada de transporte é aceitar dados da camada acima dela, dividi-los em unidades menores, se for preciso, repassar essas unidades à camada de rede e garantir que todos os fragmentos chegarão corretamente à outra extremidade. Além do mais, tudo isso deve ser feito com eficiência e de forma que as camadas superiores fiquem isoladas das inevitáveis mudanças na tecnologia de hardware com o passar do tempo.

A camada de transporte também determina que tipo de serviço deve ser fornecido à camada de sessão e, por fim, aos usuários da rede. O tipo mais popular de conexão de transporte é um canal ponto a ponto livre de erros que entrega mensagens ou bytes na ordem em que eles foram enviados. No entanto, outros possíveis tipos de serviço de transporte são as mensagens isoladas sem nenhuma garantia relativa à ordem de entrega e à propagação de mensagens para múltiplos destinos. O tipo de serviço é determinado quando a conexão é estabelecida. (Observe que é impossível conseguir um canal livre de erros; o que as pessoas realmente entendem por essa expressão é que a taxa de erros é baixa o suficiente para ser ignorada na prática.)

A camada de transporte é uma verdadeira camada de ponta a ponta, que liga a origem ao destino. Em outras palavras, um programa na máquina de origem mantém uma conversação com um programa semelhante instalado na máquina de destino, utilizando os cabeçalhos de mensagens e as mensagens de controle. Nas camadas inferiores, os protocolos são trocados entre cada uma das máquinas e seus vizinhos imediatos, e não entre as máquinas de origem e de destino, que podem estar separadas por muitos roteadores. A diferença entre as camadas 1 a 3, que são encadeadas, e as camadas 4 a 7, que são camadas de ponta a ponta, é ilustrada na Figura 1.17.

A camada de sessão

A camada de sessão permite que os usuários em diferentes máquinas estabeleçam **sessões de comunicação** entre eles. Uma sessão oferece diversos serviços, inclusive o **controle de diálogo** (mantendo o controle de quem deve transmitir em cada momento), o **gerenciamento de tokens** (impedindo que duas partes tentem executar a mesma operação crítica ao mesmo tempo) e a **sincronização** (realizando a verificação periódica de longas transmissões para permitir que elas continuem a partir do ponto em que estavam ao ocorrer uma falha e a subsequente recuperação).

A camada de apresentação

Diferente das camadas mais baixas, que se preocupam principalmente com a movimentação de bits, a **camada de apresentação** está relacionada à sintaxe e à semântica das informações transmitidas. Para tornar possível a comunicação entre computadores com diferentes representações

internas dos dados, as estruturas de dados a serem trocadas podem ser definidas de maneira abstrata, com uma codificação padrão que será usada durante a conexão. A camada de apresentação gerencia essas estruturas de dados abstratas e permite a definição e o intercâmbio de estruturas de dados de nível mais alto (por exemplo, registros bancários).

A camada de aplicação

A camada de aplicação contém uma série de protocolos comumente necessários para os usuários. Um protocolo de aplicação amplamente utilizado é o **HTTP** (**HyperText Transfer Protocol**), que constitui a base da World Wide Web. Quando um navegador deseja uma página Web, ele envia o nome da página desejada ao servidor que hospeda a página, utilizando o HTTP. O servidor, então, transmite a página ao navegador. Outros protocolos de aplicação são usados para transferências de arquivos, correio eletrônico e transmissão de notícias pela rede.

1.4.2 O modelo de referência TCP/IP

Vamos deixar de lado o modelo de referência OSI e passar ao modelo de referência usado na 'avó' de todas as redes de computadores a longa distância, a ARPANET, e sua sucessora, a Internet mundial. Embora tenhamos deixado para depois a apresentação da história da ARPANET, agora será de grande utilidade entender alguns de seus principais aspectos. A ARPANET era uma rede de pesquisa patrocinada pelo Departamento de Defesa dos Estados Unidos (DoD). Pouco a pouco, centenas de universidades e repartições públicas foram conectadas, usando linhas telefônicas dedicadas. Mais tarde, quando foram criadas as redes de rádio e satélite, os protocolos existentes começaram a ter problemas de interligação com elas, o que forçou a criação de uma nova arquitetura de referência. Desse modo, quase desde o início, a capacidade para conectar várias redes de maneira uniforme foi um dos principais objetivos do projeto. Essa arquitetura posteriormente ficou conhecida como **modelo de referência TCP/IP**, graças a seus dois principais protocolos. Esse modelo foi definido pela primeira vez em Cerf e Kahn (1974), depois melhorado e definido como um padrão na comunidade da Internet (Braden, 1989). A filosofia de projeto na qual se baseia o modelo é discutida em Clark (1988).

Diante da preocupação do Departamento de Defesa dos Estados Unidos de que seus preciosos hosts, roteadores e gateways de interconexão de redes fossem destruídos de uma hora para outra por um ataque da então União Soviética, outro objetivo principal foi que a rede fosse ser capaz de sobreviver à perda do hardware de sub-redes, sem que as conversações existentes fossem interrompidas. Em outras palavras, o Departamento de Defesa queria que as conexões permanecessem intactas enquanto as máquinas de origem e de destino estivessem funcionando, mesmo que algumas máquinas ou linhas de transmissão intermediárias deixassem de operar repentinamente. Além disso, como eram visadas aplicações com requisitos divergentes, desde a transferência de arquivos e a transmissão de dados de voz em tempo real, era necessária uma arquitetura flexível.

A camada de enlace

Todas essas necessidades levaram à escolha de uma rede de comutação de pacotes baseada em uma camada de interligação de redes com serviço não orientado a conexões, passando por diferentes topologias de redes. A **camada de enlace**, a mais baixa no modelo, descreve o que os enlaces como linhas seriais e a Ethernet clássica precisam fazer para cumprir os requisitos dessa camada de interconexão com serviço não orientado a conexões. Ela não é uma camada propriamente dita, no sentido normal do termo, mas uma interface entre os hosts e os enlaces de transmissão. O material inicial sobre o modelo TCP/IP tem pouco a dizer sobre ela.

A camada internet (camada de rede)

A **camada internet** integra toda a arquitetura, mantendo-a unida. Ela aparece na Figura 1.18 como uma correspondência aproximada com o modelo de rede OSI. Sua tarefa é permitir que os hosts injetem pacotes em qualquer rede e garantir que eles trafegarão independentemente até o destino (talvez em uma rede diferente). Eles podem che-

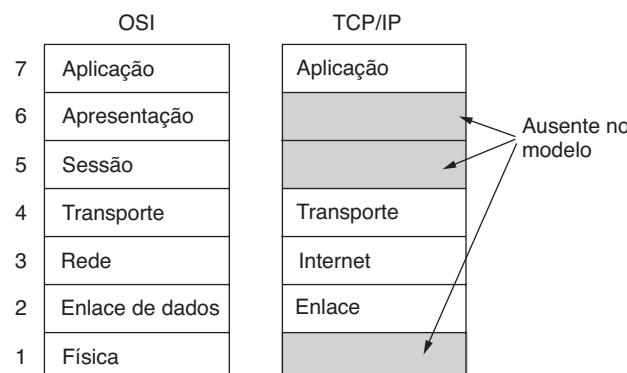



Figura 1.18 | O modelo de referência TCP/IP.

gar até mesmo em uma ordem diferente daquela em que foram enviados, obrigando as camadas superiores a reorganizá-los, caso a entrega em ordem seja desejável. Observe que o termo internet (rede interligada) é usado aqui em um sentido genérico, embora essa camada esteja presente na Internet.

A analogia usada nesse caso diz respeito ao sistema de correio (convencional). Uma pessoa pode deixar uma sequência de cartas internacionais em uma caixa de correio em um país e, com um pouco de sorte, a maioria delas será entregue no endereço correto no país de destino. Provavelmente, as cartas atravessarão um ou mais centros de triagem de correio internacionais ao longo do caminho, mas isso é transparente para os usuários. Além disso, o fato de cada país (ou seja, cada rede) ter seus próprios selos, tamanhos de envelope preferidos e regras de entrega fica oculto dos usuários.

A camada internet define um formato de pacote oficial e um protocolo chamado **IP** (**Internet Protocol**), mais um protocolo que o acompanha, chamado ICMP (**Internet Control Message Protocol**). A tarefa da camada internet é entregar pacotes IP onde eles são necessários. O roteamento de pacotes claramente é uma questão de grande importância nessa camada, assim como o congestionamento (embora o IP não seja eficaz para evitar o congestionamento).

A camada de transporte

No modelo TCP/IP, a camada localizada acima da camada internet agora é chamada **camada de transporte**. A finalidade dessa camada é permitir que as entidades pares dos hosts de origem e de destino mantenham uma conversação, exatamente como acontece na camada de transporte OSI. Dois protocolos de ponta a ponta foram definidos aqui. O primeiro deles, o protocolo de controle de transmissão, ou **TCP** (**Transmission Control Protocol**), é um protocolo orientado a conexões confiável que permite a entrega sem erros de um fluxo de bytes originário de uma determinada máquina em qualquer computador da internet. Esse protocolo fragmenta o fluxo de bytes de entrada em mensagens discretas e passa cada uma delas para a camada internet. No destino, o processo TCP receptor volta a montar as mensagens recebidas no fluxo de saída. O TCP também cuida do controle de fluxo, impedindo que um transmissor rápido sobrecarregue um receptor lento com um volume de mensagens maior do que ele pode manipular.

O segundo protocolo nessa camada, o protocolo de datagrama do usuário, ou **UDP** (**User Datagram Protocol**), é um protocolo sem conexões, não confiável, para aplicações que não desejam a sequência ou o controle de fluxo do TCP, e que desejam oferecer seu próprio controle. Ele é muito usado para consultas isoladas, com solicitação e resposta, tipo cliente-servidor, e aplicações em que a entrega imediata é mais importante do que a entrega precisa, como na transmissão de voz ou vídeo. A relação entre IP, TCP e UDP é ilustrada na Figura 1.19. Desde que o modelo foi desenvolvido, o IP tem sido implementado em muitas outras redes.

A camada de aplicação

O modelo TCP/IP não tem as camadas de sessão ou de apresentação. Não foi percebida qualquer necessidade para elas. Ao invés disso, as aplicações simplesmente incluem quaisquer funções de sessão e apresentação que forem necessárias. A experiência com o modelo OSI demonstrou que essa visão está correta: elas são pouco usadas na maioria das aplicações.

Acima da camada de transporte, encontramos a **camada de aplicação**. Ela contém todos os protocolos de nível mais alto. Dentre eles estão o protocolo de terminal virtual (TELNET), o protocolo de transferência de arquivos (FTP) e o protocolo de correio eletrônico (SMTP). Muitos outros protocolos foram incluídos no decorrer dos anos. Alguns dos mais importantes que estudaremos, mostrados na Figura 1.19, incluem o DNS (Domain Name Service), que mapeia os nomes de hosts para seus respectivos endereços da camada de rede (Internet), o HTTP, protocolo usado para buscar páginas na World Wide Web, e o RTP, protocolo para entregar mídia em tempo real, como voz ou vídeo.

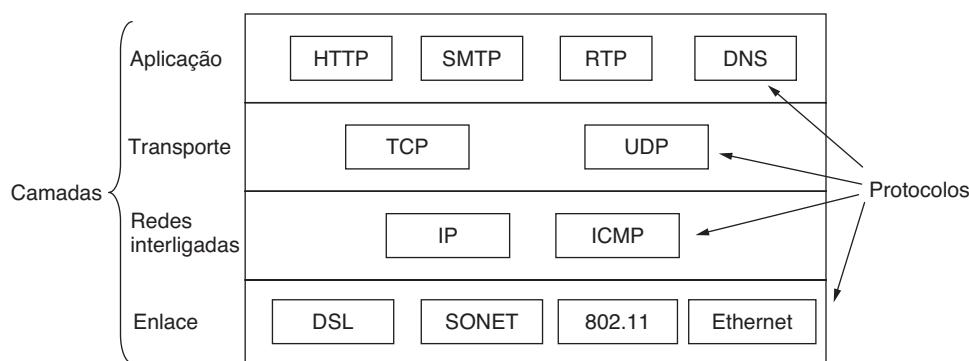



Figura 1.19 | O modelo TCP/IP com alguns protocolos que estudaremos.

1.4.3 O modelo de dados usado neste livro

Conforme dissemos, o ponto forte do modelo de referência OSI é o *modelo* propriamente dito (menos as camadas de apresentação e sessão), que provou ser excepcionalmente útil para a discussão de redes de computadores. Por outro lado, o ponto forte do modelo de referência TCP/IP são os *protocolos*, que têm sido bastante utilizados há muitos anos. Como os cientistas da computação gostam de ter seu bolo e comê-lo também, usaremos o modelo híbrido da Figura 1.20 como base para este livro.

Esse modelo tem cinco camadas, partindo da camada física e subindo pelas camadas de enlace, rede e transporte, até chegar à camada de aplicação. A camada física especifica como transmitir os bits por diferentes tipos de mídia como sinais elétricos (ou outro semelhante). A camada de enlace trata de como enviar mensagens de tamanho definido entre computadores diretamente conectados, com níveis de confiabilidade especificados. Ethernet e 802.11 são exemplos de padrões da camada de enlace.

A camada de rede cuida de como combinar vários enlaces nas redes, e redes de redes em internets, de modo a enviar pacotes entre computadores distantes. Isso inclui a tarefa de localizar o caminho pelo qual os pacotes serão enviados. O IP é o principal exemplo de protocolo que estudaremos para essa camada. A camada de transporte fortalece as garantias de entrega da camada de rede, normalmente com maior confiabilidade, e oferece abstrações de entrega, como um fluxo de bytes confiável, que correspondem às necessidades das diferentes aplicações. O TCP é um exemplo importante de protocolo da camada de transporte.

Por fim, a camada de aplicação contém programas que utilizam a rede. Muitas aplicações de rede, mas não todas, possuem interfaces com o usuário, como um navegador Web. Contudo, nossa preocupação é com a parte do programa que usa a rede. No caso do navegador Web, esse é o protocolo HTTP. Também existem programas de suporte importantes na camada de aplicação, como o DNS, que são usados por muitas aplicações.

Nossa sequência de capítulos é baseada nesse modelo. Dessa forma, mantemos o valor do modelo OSI para entender as arquiteturas de rede, mas nos concentramos principalmente nos protocolos que são importantes na prática, do TCP/IP e protocolos relacionados, aos mais novos como os padrões 802.11, SONET e Bluetooth.

Figura 1.20 | O modelo de referência usado neste livro.

1.4.4 Uma comparação entre os modelos de referência OSI e TCP/IP

Os modelos de referência OSI e TCP/IP têm muito em comum. Ambos se baseiam no conceito de uma pilha de protocolos independentes. Além disso, as camadas têm praticamente as mesmas funcionalidades. Por exemplo, em ambos os modelos estão presentes as camadas que englobam até a camada de transporte para oferecer um serviço de transporte de ponta a ponta, independente da rede, a processos que desejam se comunicar. Essas camadas formam o provedor de transporte. Mais uma vez, em ambos os modelos, as camadas acima da camada de transporte dizem respeito às aplicações que fazem uso do serviço de transporte.

Apesar dessas semelhanças fundamentais, os dois modelos também têm muitas diferenças. Nesta seção, vamos nos deter nas principais diferenças existentes entre os dois modelos de referência. É importante notar que estamos comparando os *modelos de referência*, não as *pilhas de protocolos* correspondentes. Os protocolos propriamente ditos serão discutidos em seguida. Para examinar as semelhanças e as diferenças entre o TCP/IP e o OSI, consulte Piscitello e Chapin (1993).

O modelo OSI tem três conceitos fundamentais:

1. Serviços.
2. Interfaces.
3. Protocolos.

Provavelmente a maior contribuição do modelo OSI seja tornar explícita a distinção entre esses três conceitos. Cada camada executa alguns *serviços* para a camada acima dela. A definição do serviço informa o que a camada faz, e não a forma como as entidades acima dela a acessam ou como a camada funciona. Essa definição estabelece a semântica da camada.

A *interface* de uma camada informa como os processos acima dela podem acessá-la. Também especifica quais são os parâmetros e os resultados a serem esperados. Ela também não revela o funcionamento interno da camada.

Finalmente, os *protocolos* utilizados em uma camada são de responsabilidade dessa camada. Esta pode usar os protocolos que quiser, desde que realize o trabalho (ou seja, forneça os serviços oferecidos). Ela também pode alterar esses protocolos sem influenciar o software das camadas superiores.

Essas ideias se adaptam perfeitamente aos novos conceitos da programação orientada a objetos. Um objeto, assim como uma camada, tem um conjunto de métodos (operações) que os processos externos ao objeto podem invocar. A semântica desses métodos define o conjunto de serviços que o objeto oferece. Os parâmetros dos métodos e os resultados deles formam a interface do objeto. O código interno do objeto é seu protocolo, que não é visível nem interessa aos elementos fora do objeto.

Originalmente, o modelo TCP/IP não distinguia com clareza a diferença entre serviços, interface e protocolo, embora as pessoas tenham tentado deixá-lo mais parecido com o modelo OSI. Por exemplo, os únicos serviços reais oferecidos pela camada de rede interligada (Internet) são send ip packet (enviar pacote IP) e receive ip packet (receber pacote IP). Por conseguinte, os protocolos no modelo OSI são mais bem encapsulados que os do modelo TCP/IP e podem ser alterados com relativa facilidade, conforme a tecnologia muda. Um dos principais objetivos das diversas camadas de protocolos é permitir a realização transparente dessas alterações.

O modelo de referência OSI foi concebido *antes* de os protocolos correspondentes terem sido criados. Isso significa que o modelo não teve influência de um determinado conjunto de protocolos, o que o deixou bastante genérico. A desvantagem dessa ordenação foi que os projetistas não tinham muita experiência no assunto nem muita noção sobre a funcionalidade que deveria ser incluída em cada camada.

Por exemplo, a camada de enlace de dados lidava originalmente com redes ponto a ponto. Quando surgiram as redes de broadcast, foi preciso criar uma nova subcamada no modelo. Além disso, quando as pessoas começaram a criar redes reais com base no modelo OSI e nos protocolos existentes, descobriu-se que essas redes não eram compatíveis com as especificações de serviço exigidas (que maravilha), de modo que foi necessário enxertar no modelo subcamadas de convergência que permitissem atenuar as diferenças. Por fim, como o comitê acreditava que cada país teria uma rede controlada pelo governo e baseada nos protocolos OSI, não se preocupou com as conexões entre as redes. Para encurtar a história: na prática, as coisas aconteceram de maneira muito diferente da teoria.

Com o TCP/IP, ocorreu exatamente o contrário: como os protocolos vieram primeiro, o modelo realmente foi criado como uma descrição dos protocolos existentes. Não houve problemas para os protocolos serem adaptados ao modelo. Eles se encaixaram perfeitamente. O único problema foi o fato de o *modelo* não se adaptar a outras pilhas de protocolos. Consequentemente, ele não tinha muita utilidade para descrever outras redes que não faziam uso do protocolo TCP/IP.

Deixando a filosofia de lado e entrando em questões mais específicas, uma diferença óbvia entre os dois modelos está no número de camadas: o modelo OSI tem sete camadas e o TCP/IP tem quatro. Ambos têm as camadas de rede, transporte e aplicação, mas as outras são diferentes.

Outra diferença está na área da comunicação não orientada a conexões *versus* comunicação orientada a conexões. Na camada de rede, o modelo OSI é compatível com a comunicação não orientada a conexões e com a comunicação orientada a conexões; no entanto, na camada de transporte, o modelo aceita apenas a comunicação orientada a conexões, onde de fato ela é mais importante (pois o serviço de transporte é visível para os usuários). O modelo TCP/IP só tem um modo de operação na camada de rede (não orientado a conexões), mas aceita ambos os modos na camada de transporte, oferecendo aos usuários a possibilidade de escolha. Essa escolha é especialmente importante para os protocolos simples de solicitação/resposta.

1.4.5 Uma crítica aos protocolos e ao modelo OSI

Nem o modelo OSI e seus respectivos protocolos nem o modelo TCP/IP e seus respectivos protocolos são perfeitos. Os dois podem ser e têm sido alvo de uma série de críticas. Nesta seção e na próxima, examinaremos algumas delas. Começaremos pelo modelo OSI e, em seguida, examinaremos o TCP/IP.

Na época em que a segunda edição americana deste livro foi publicada (1989), muitos especialistas tinham a impressão de que os protocolos e o modelo OSI controlariam o mundo e atropelariam tudo o que se pusesse em seu caminho. Isso não aconteceu. Por quê? Vale a pena fazer uma revisão de algumas razões, que podem ser resumidas da seguinte maneira:

1. Momento ruim.
2. Tecnologia ruim.
3. Implementações ruins.
4. Política ruim.

Momento ruim

Vamos começar pelo problema mais importante: momento ruim. O momento em que um padrão é estabelecido é de fundamental importância para seu sucesso. David Clark, do MIT, tem uma teoria sobre os padrões, que ele chama *o apocalipse dos dois elefantes*, ilustrada na Figura 1.21.

Essa figura mostra o volume de atividades relacionadas a um novo assunto. Quando o assunto é descoberto, há uma grande atividade de pesquisa, em forma de discussões, artigos e reuniões. Após algum tempo dessa atividade inicial, as empresas descobrem o assunto e tem início a onda de bilhões de dólares em investimentos.

É essencial que os padrões sejam desenvolvidos entre os dois 'elefantes'. Se eles forem desenvolvidos muito cedo, antes de a pesquisa ser concluída, o assunto poderá não estar devidamente compreendido; o resultado é um padrão ruim. Se eles forem desenvolvidos muito tarde, muitas empresas talvez já tenham feito investimentos maciços para descobrir maneiras diferentes de tirar proveito dessa nova tecnologia e, portanto, os padrões serão efetivamente ignorados. Se o intervalo entre os dois elefantes for muito curto (porque todos estão apressados para começar), a equipe de desenvolvimento dos padrões poderá ser esmagada.

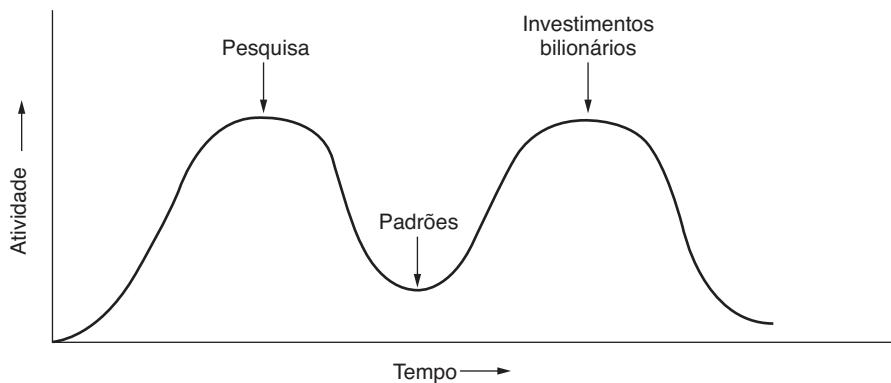



Figura 1.21 | O apocalipse dos dois elefantes.

Hoje se sabe que os protocolos do padrão OSI foram esmagados. Os protocolos TCP/IP concorrentes já estavam sendo amplamente utilizados nas universidades de pesquisa na época em que apareceram os protocolos OSI. Antes mesmo do início da onda de investimentos de bilhões de dólares, o mercado acadêmico já era suficientemente grande, e muitos fabricantes começaram a oferecer produtos TCP/IP, apesar de estarem cautelosos. Quando o OSI surgiu, eles não estavam dispostos a investir em uma segunda pilha de protocolos enquanto não fossem forçados a isso, daí não ter havido ofertas iniciais no mercado. Com todas as empresas aguardando que alguém desse o primeiro passo, nenhuma empresa o iniciou, e o OSI nunca aconteceu.

Tecnologia ruim

A segunda razão para que o OSI não vingasse estava nas falhas do modelo e dos protocolos. A escolha de sete camadas foi mais política do que técnica, e duas camadas (a de sessão e a de apresentação) estão praticamente vazias, enquanto duas outras (a de enlace de dados e a de rede) se encontram sobrecarregadas.

O modelo OSI, com os protocolos e as definições de serviços inter-relacionados, é extraordinariamente complexo. Quando empilhados, os padrões impressos chegam a quase um metro de altura. Além disso, eles são de difícil implementação e sua operação não é nada eficiente. Nesse contexto, vale a pena lembrar o enigma proposto por Paul Mockapetris e citado em Rose (1993):

> **P:** O que você vê quando encontra um mafioso que adota um padrão internacional?
>
> **R:** Alguém que lhe faz uma oferta que você não consegue entender.

Além de ser incompreensível, outro problema com o OSI é que algumas funções, como endereçamento e controle de fluxo e de erros, aparecem repetidamente em cada camada. Por exemplo, Saltzer et al. (1984) lembraram que, para ser eficaz, o controle de erros deve ser feito na camada mais alta, de modo que sua repetição em cada uma das camadas inferiores é desnecessária e ineficiente.

Implementações ruins

Em virtude da enorme complexidade do modelo e dos protocolos, ninguém ficou surpreso com o fato de as implementações iniciais serem lentas, pesadas e gigantescas. Todas as pessoas que as experimentaram saíram chamuscadas. Não demorou muito para que elas associassem 'OSI' a 'baixa qualidade'. A imagem resistiu inclusive às significativas melhorias a que os produtos foram submetidos com o decorrer do tempo.

Por outro lado, uma das primeiras implementações do TCP/IP fazia parte do UNIX de Berkeley e era muito boa (sem contar que era gratuita). As pessoas começaram a usá-la rapidamente, criando, assim, uma grande comunidade de usuários que, por sua vez, estimulou novas melhorias, que levaram a uma comunidade ainda maior. Nesse caso, a espiral foi claramente ascendente, não descendente.

Política ruim

Em decorrência da implementação inicial, muitas pessoas, em particular no universo acadêmico, pensaram que o TCP/IP era parte do UNIX e, na década de 1980, as universidades tinham verdadeira adoração pelo UNIX.

Por outro lado, o OSI era considerado uma criação dos ministérios de telecomunicações europeus, da Comunidade Europeia e, mais tarde, do governo dos Estados Unidos. Essa crença só era verdadeira em parte, mas a ideia de um punhado de burocratas tentando empurrar um padrão tecnicamente inferior pela garganta dos pobres pesquisadores e programadores que de fato trabalhavam no desenvolvimento de redes de computadores não foi de muita ajuda à causa do OSI. Algumas pessoas viram nesse desenvolvimento a repetição de um episódio da década de 1960, quando a IBM anunciou que a PL/I era a linguagem do futuro; mais tarde, essa afirmação foi desmentida pelo Departamento de Defesa dos Estados Unidos, que afirmou que a linguagem do futuro seria a Ada.

1.4.6 Uma crítica ao modelo de referência TCP/IP

Os protocolos e o modelo TCP/IP também tiveram os seus problemas. Em primeiro lugar, o modelo não diferencia com a clareza necessária os conceitos de serviço, interface e protocolo. A boa prática da engenharia de software exige uma diferenciação entre especificação e implementação, algo que o OSI faz com muito cuidado, ao contrário do TCP/IP. Consequentemente, o modelo TCP/IP não é o melhor dos guias para a criação de novas redes com base em novas tecnologias.

Em segundo lugar, o modelo TCP/IP não é nada abrangente e não consegue descrever outras pilhas de protocolos que não a pilha TCP/IP. Por exemplo, seria praticamente impossível tentar descrever o Bluetooth usando esse modelo.

Terceiro, a camada host/rede não é realmente uma camada no sentido em que o termo é usado no contexto dos protocolos hierarquizados. Trata-se, na verdade, de uma interface (entre as camadas de rede e de enlace de dados). A distinção entre uma interface e uma camada é crucial e você deve considerá-la com cuidado.

Em quarto lugar, o modelo TCP/IP não faz distinção entre as camadas física e de enlace de dados. Elas são completamente diferentes. A camada física está relacionada às características de transmissão de fio de cobre, dos cabos de fibra óptica e da comunicação sem fio. A tarefa da camada de enlace de dados é delimitar o início e o final dos quadros e enviá-los de um lado a outro com o grau de confiabilidade desejado. Um modelo mais adequado deve incluir as duas camadas como elementos distintos. O modelo TCP/IP não faz isso.

Por fim, apesar de os protocolos IP e TCP terem sido cuidadosamente projetados e bem implementados, o mesmo não aconteceu com muitos outros protocolos ocasionais, geralmente produzidos por alguns alunos formados, pesquisando até se cansarem. As implementações desses protocolos eram distribuídas gratuitamente, o que acabava difundindo seu uso de tal forma que se tornou difícil substituí-las. Hoje em dia, a fidelidade a esses produtos é motivo de alguns embaraços. Por exemplo, o protocolo de terminal virtual, o TELNET, foi projetado para um terminal TTY mecânico, capaz de processar dez caracteres por segundo. Ele não reconhece o mouse e as interfaces gráficas do usuário. No entanto, esse protocolo é usado em larga escala ainda hoje, trinta anos depois de seu surgimento.

1.5 Exemplos de redes

O assunto de redes de computadores abrange muitos tipos diferentes de redes, grandes e pequenas, bem conhecidas e pouco conhecidas. Elas têm diferentes objetivos, escalas e tecnologias. Nas seções a seguir, examinaremos alguns exemplos, para termos uma ideia da variedade existente na área de redes de computadores.

Começaremos com a Internet, provavelmente a rede mais conhecida, e estudaremos sua história, sua evolução e sua tecnologia. Em seguida, consideraremos a rede de telefonia móvel. Tecnicamente, ela é muito diferente da Internet, havendo um bom contraste entre ambas. Depois, veremos o IEEE 802.11, o padrão dominante para LANs sem fios. Finalmente, examinaremos a RFID e redes de sensores, tecnologias que estendem o alcance da rede para incluir o mundo físico e objetos cotidianos.

1.5.1 A Internet

A Internet não é de modo algum uma rede, mas sim um vasto conjunto de redes diferentes que utilizam certos protocolos comuns e fornecem determinados serviços comuns. É um sistema incomum no sentido de não ter sido planejado nem ser controlado por ninguém. Para entendê-la melhor, vamos começar do início e observar como e por que ela foi desenvolvida. Se desejar conhecer uma história maravilhosa sobre o surgimento da Internet, recomendamos o livro de John Naughton (2000). Trata-se de um daqueles raros livros que não apenas são divertidos de ler, mas que também têm 20 páginas de citações destinadas aos historiadores sérios. Uma parte do material a seguir se baseia nesse livro.

É claro que também foram escritos inúmeros livros técnicos sobre a Internet e seus protocolos. Para obter mais informações consulte, por exemplo, Maufer (1999).

A ARPANET

A história começa no final da década de 1950. No auge da Guerra Fria, o Departamento de Defesa dos Estados Unidos queria uma rede de controle e comando capaz de sobreviver a uma guerra nuclear. Nessa época, todas as comunicações militares passavam pela rede de telefonia pública, considerada vulnerável. A razão para essa convicção pode ser vista na Figura 1.22(a). Nessa figura, os pontos pretos representam centrais de comutação telefônica, cada uma das quais conectada a milhares de telefones. Por sua vez, essas centrais de comutação estavam conectadas a centrais de comutação de nível mais alto (centrais interurbanas), formando uma hierarquia nacional com apenas uma pequena redundância. A vulnerabilidade do sistema era o fato de que a destruição de algumas centrais interurbanas importantes poderia fragmentar o sistema em muitas ilhas isoladas.

Por volta de 1960, o Departamento de Defesa dos Estados Unidos firmou um contrato com a RAND Corporation para encontrar uma solução. Um de seus funcionários, Paul Baran, apresentou o projeto altamente distribuído e tolerante a falhas da Figura 1.22(b). Tendo em vista que os caminhos entre duas centrais de comutação quaisquer

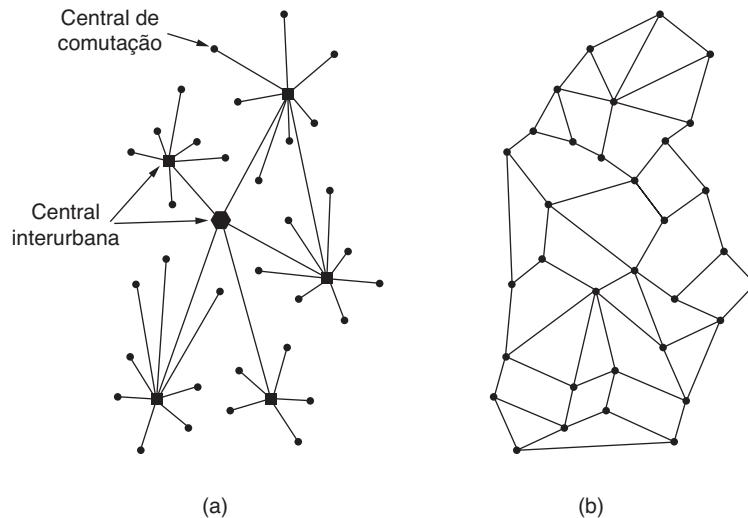



Figura 1.22 | (a) Estrutura do sistema de telefonia. (b) Sistema distribuído de comutação proposto por Baran.

eram agora muito mais longos do que a distância que os sinais analógicos podiam percorrer sem distorção, Baran propôs o uso da tecnologia digital de comutação de pacotes. Ele enviou diversos relatórios para o Departamento de Defesa descrevendo suas ideias em detalhes (Baran, 1964). Os funcionários do Pentágono gostaram do conceito e pediram à AT&T, na época a empresa que detinha o monópolio nacional da telefonia nos Estados Unidos, que construísse um protótipo. A AT&T descartou as ideias de Baran. Afinal, a maior e mais rica corporação do mundo não podia permitir que um jovem pretensioso lhe ensinasse a criar um sistema telefônico. A empresa informou que a rede de Baran não podia ser construída e a ideia foi abandonada.

Vários anos se passaram e o Departamento de Defesa ainda não tinha um sistema melhor de comando e controle. Para entender o que aconteceu em seguida, temos de retornar a outubro de 1957, quando a União Soviética derrotou os Estados Unidos na corrida espacial com o lançamento do primeiro satélite artificial, o Sputnik. Quando tentou descobrir quem tinha 'dormido no ponto', o presidente Eisenhower acabou detectando a disputa entre o Exército, a Marinha e a Força Aérea pelo orçamento de pesquisa do Pentágono. Sua resposta imediata foi criar uma organização centralizada de pesquisa de defesa, a **ARPA**, ou **Advanced Research Projects Agency**. A ARPA não tinha cientistas nem laboratórios; de fato, ela não tinha nada além de um escritório e um pequeno orçamento (para os padrões do Pentágono). A agência realizava seu trabalho oferecendo concessões e contratos a universidades e empresas cujas ideias lhe pareciam promissoras.

Durante os primeiros anos, a ARPA tentou compreender qual deveria ser sua missão. Porém, em 1967, a atenção do então diretor de programas da ARPA, Larry Roberts, que estava tentando descobrir como oferecer acesso remoto aos computadores, se voltou para as redes. Ele entrou em contato com diversos especialistas para decidir o que fazer. Um deles, Wesley Clark, sugeriu a criação de uma sub-rede comutada por pacotes, dando a cada host seu próprio roteador.

Após algum ceticismo inicial, Roberts comprou a ideia e apresentou um documento um tanto vago sobre ela no ACM SIGOPS Symposium on Operating System Principles, realizado em Gatlinburg, Tennessee, no final de 1967 (Roberts, 1967). Para grande surpresa de Roberts, outro documento na conferência descrevia um sistema semelhante, que não só tinha sido projetado mas, na realidade, havia sido totalmente implementado sob a orientação de Donald Davies no National Physical Laboratory, na Inglaterra. O sistema do NPL não era um sistema nacional (ele simplesmente conectava vários computadores no campus do NPL), mas demonstrava que a comutação de pacotes podia funcionar. Além disso, ele citava o trabalho anteriormente descartado de Baran. Roberts voltou de Gatlinburg determinado a construir o que mais tarde ficou conhecido como **ARPANET**.

A sub-rede consistiria em minicomputadores chamados processadores de mensagens de interface, ou **IMPs** (**Interface Message Processors**), conectados por linhas de transmissão de 56 kbps. Para garantir sua alta confiabilidade, cada IMP seria conectado a pelo menos dois outros IMPs. A sub-rede tinha de ser uma sub-rede de datagrama, de modo que, se algumas linhas e alguns IMPs fossem destruídos, as mensagens poderiam ser roteadas automaticamente para caminhos alternativos.

Cada nó da rede deveria ter um IMP e um host na mesma sala, conectados por um fio curto. Um host poderia enviar mensagens de até 8.063 bits para seu IMP que, em seguida, dividiria essas mensagens em pacotes de no máximo 1.008 bits e os encaminharia de forma independente até o destino. Cada pacote era recebido por completo antes

de ser encaminhado; assim, a sub-rede se tornou a primeira rede eletrônica de comutação de pacotes de store-and--forward (de armazenamento e encaminhamento).

Em seguida, a ARPA abriu uma concorrência para a construção da sub-rede. Doze empresas apresentaram propostas. Depois de avaliar todas as propostas, a ARPA selecionou a BBN, uma empresa de consultoria de Cambridge, Massachusetts e, em dezembro de 1968, assinou um contrato para montar a sub-rede e desenvolver o software para ela. A BBN resolveu utilizar, como IMPs, minicomputadores Honeywell DDP-316 especialmente modificados, com 12 K palavras de 16 bits de memória principal. Os IMPs não tinham unidades de discos, pois os componentes móveis eram considerados pouco confiáveis. Os IMPs eram interconectados por linhas privadas das companhias telefônicas, de 56 kbps. Embora 56 kbps hoje seja a única escolha de adolescentes que não podem dispor de DSL ou modems a cabo, na época era o melhor que o dinheiro podia comprar.

O software foi dividido em duas partes: sub-rede e host. O software da sub-rede consistia na extremidade IMP da conexão host-IMP, no protocolo IMP-IMP e em um protocolo do IMP de origem para o IMP de destino, criado para aumentar a confiabilidade. O projeto original da ARPANET é mostrado na Figura 1.23.

Fora da sub-rede, também havia necessidade de software, ou seja, a extremidade referente ao host da conexão host-IMP, o protocolo host-host e o software de aplicação. Logo ficou claro que a BBN era da opinião que, quando tivesse aceitado uma mensagem em uma conexão host-IMP e a tivesse colocado na conexão host-IMP no destino, sua tarefa teria terminado.

Entretanto, Roberts tinha um problema: os hosts também precisavam de software. Para lidar com ele, Roberts convocou uma reunião com os pesquisadores de rede, em sua maioria estudantes universitários, em Snowbird, Utah, no verão de 1969. Os universitários esperavam que algum perito em redes explicasse o projeto geral da rede e seu software, e depois atribuísse a cada um deles a tarefa de desenvolver uma parte do projeto. Eles ficaram absolutamente surpresos ao verem que não havia nenhum especialista em rede e nenhum projeto geral. Os estudantes teriam de descobrir o que fazer por conta própria.

No entanto, em dezembro de 1969 entrou no ar uma rede experimental com quatro nós: UCLA, UCSB, SRI e University of Utah. Esses quatro nós foram escolhidos porque todos tinham um grande número de contratos com a ARPA, e todos tinham computadores host diferentes e completamente incompatíveis (para aumentar o desafio). A primeira mensagem host a host havia sido enviada dois meses antes, do nó na UCLA, por uma equipe liderada por Len Kleinrock (pioneiro da teoria de comutação de pacotes) para o nó em SRI. A rede cresceu rapidamente à medida que outros IMPs foram entregues e instalados; logo se estendeu por todo o território norte-americano. A Figura 1.24 mostra a rapidez com que a ARPANET se desenvolveu nos três primeiros anos.

Além de ajudar no súbito crescimento da ARPANET, a ARPA também financiou pesquisas sobre o uso de redes de satélite e redes móveis de rádio de pacotes. Em uma hoje famosa demonstração, um motorista de caminhão viajando pela Califórnia utilizou a rede de rádio de pacotes para enviar mensagens à SRI, que então foram encaminhadas pela ARPANET até a Costa Leste dos Estados Unidos, de onde foram enviadas à University College, em Londres, pela rede de satélite. Isso permitiu que um pesquisador no caminhão usasse um computador situado em Londres enquanto dirigia pelo estado da Califórnia.

Essa experiência também demonstrou que os protocolos da ARPANET não eram adequados para execução em redes diferentes. Essa observação levou a mais pesquisas sobre protocolos, culminando com a invenção dos protocolos e do modelo TCP/IP (Cerf e Kahn, 1974). O TCP/IP foi criado especificamente para manipular a comunicação entre redes interligadas, algo que se tornou mais importante à medida que um número maior de redes era conectado à ARPANET.

Para estimular a adoção desses novos protocolos, a ARPA ofereceu diversos contratos para implementar o TCP/IP em diferentes plataformas de computação, incluindo sistemas IBM, DEC e HP, bem como o UNIX de Berkeley. Os

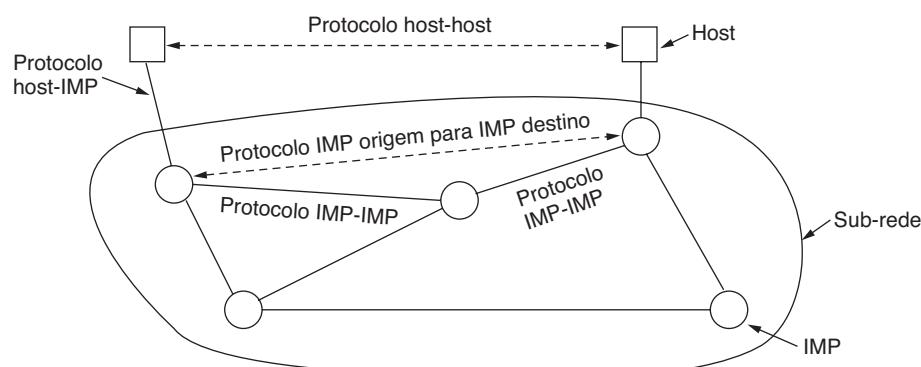



Figura 1.23 | O projeto original da ARPANET.

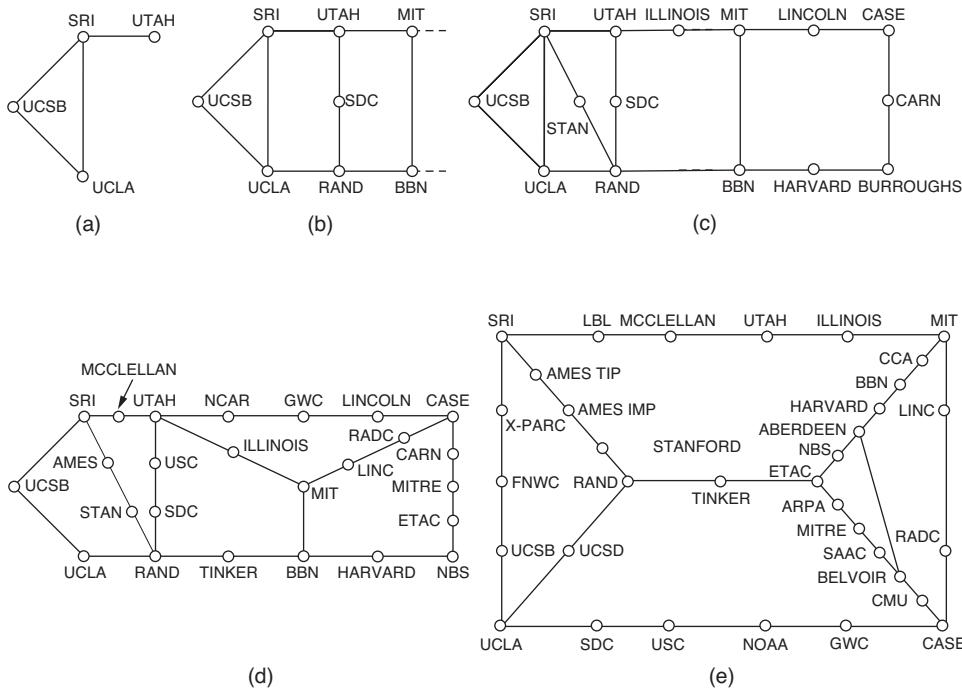



Figura 1.24 | O crescimento da ARPANET. (a) Dezembro de 1969. (b) Julho de 1970. (c) Março de 1971. (d) Abril de 1972. (e) Setembro de 1972.

pesquisadores na University of California em Berkeley reescreveram o TCP/IP com uma nova interface de programação (**soquetes**) para o lançamento iminente da versão 4.2BSD do UNIX de Berkeley. Eles também escreveram muitos programas aplicativos, utilitários e de gerenciamento para mostrar como era conveniente usar a rede com soquetes.

A ocasião foi perfeita. Muitas universidades tinham acabado de adquirir um segundo ou um terceiro computador VAX e uma LAN para conectá-los, mas não tinham nenhum software de rede. Quando surgiu o 4.2BSD, com TCP/IP, soquetes e muitos utilitários de rede, o pacote completo foi adotado imediatamente. Além disso, com o TCP/IP, era fácil conectar as LANs à ARPANET, e muitos fizeram isso.

Durante a década de 1980, novas redes, em particular as LANs, foram conectadas à ARPANET. À medida que a escala aumentou, tornou-se cada vez mais dispendioso localizar os hosts, e assim foi criado o sistema de nomes de domínio, ou **DNS** (**Domain Name System**), para organizar máquinas em domínios e relacionar nomes de hosts com endereços IP. Desde então, o DNS se transformou em um sistema generalizado de bancos de dados distribuídos, capaz de armazenar uma série de informações referentes à atribuição de nomes. Estudaremos esse sistema em detalhes no Capítulo 7.

NSFNET

No final da década de 1970, a NSF (National Science Foundation) percebeu o enorme impacto que a ARPANET estava causando nas pesquisas universitárias nos Estados Unidos, permitindo que cientistas de todo o país compartilhassem dados e trabalhassem juntos em projetos de pesquisa. No entanto, para entrar na ARPANET, uma universidade precisava ter um contrato de pesquisa com o Departamento de Defesa dos Estados Unidos, e muitas não o tinham. A resposta inicial da NSF foi patrocinar a Computer Science Network (**CSNET**) em 1981. Ela conectava departamentos de ciência da computação e laboratórios de pesquisa industrial à ARPANET por meio de linhas discadas e privadas. No final da década de 1980, a NSF foi ainda mais longe e decidiu desenvolver uma sucessora para a ARPANET, que seria aberta a todos os grupos de pesquisa universitários.

Para ter algo concreto com que começar, a NSF decidiu construir uma rede de backbone para conectar seus seis centros de supercomputadores, localizados em San Diego, Boulder, Champaign, Pittsburgh, Ithaca e Princeton. Cada supercomputador ganhou um irmão mais novo, um microcomputador LSI-11, chamado **fuzzball**. Os fuzzballs estavam conectados a linhas privadas de 56 kbps e formavam a sub-rede, usando a mesma tecnologia de hardware da ARPANET. Porém, a tecnologia de software era diferente: os fuzzballs se comunicavam diretamente com o TCP/IP desde o início, criando, assim, a primeira WAN TCP/IP.

A NSF também financiou cerca de vinte redes regionais que foram conectadas ao backbone para permitir que os usuários de milhares de universidades, laboratórios de pesquisa, bibliotecas e museus tivessem acesso a um dos supercomputadores e se comunicassem entre si. A rede completa, incluindo o backbone e as redes regionais, foi

chamada **NSFNET**. Ela se conectava à ARPANET por meio de um link entre um IMP e um fuzzball na central de processamento de dados da Carnegie-Mellon. O primeiro backbone da NSFNET está ilustrado na Figura 1.25, sobreposta a um mapa dos Estados Unidos.

A NSFNET foi um sucesso instantâneo e logo estava sobrecarregada. Imediatamente, a NSF começou a planejar sua sucessora e firmou um contrato com o consórcio MERIT, de Michigan, para executá-la. Junto à MCI foram alugados canais de fibra óptica de 448 kbps (na época absorvida pela WorldCom) para fornecer a versão 2 do backbone. Máquinas IBM PC-RT foram usadas como roteadores. Logo o segundo backbone também estava operando com sua capacidade máxima e, em 1990, ele foi atualizado para 1,5 Mbps.

O contínuo crescimento levou a NSF a perceber que o governo não podia continuar a financiar a rede para sempre. Além disso, as organizações comerciais queriam participar da rede, mas eram proibidas pelo estatuto da NSF de utilizar redes mantidas com verbas da fundação. Consequentemente, a NSF estimulou a MERIT, a MCI e a IBM a formarem uma empresa sem fins lucrativos, a **ANS** (**Advanced Networks and Services**) que, na prática, foi a primeira etapa em direção à comercialização. Em 1990, a ANS assumiu a NSFNET e atualizou os links de 1,5 Mbps para 45 Mbps, a fim de formar a **ANSNET**. Essa rede operou por cinco anos e depois foi vendida à America Online. Porém, nessa época, diversas empresas estavam oferecendo o serviço IP comercial e se tornou claro que o governo deveria deixar o negócio de redes.

Para facilitar a transição e garantir que todas as redes regionais pudessem se comunicar entre si, a NSF contratou quatro diferentes operadoras de redes para estabelecer um ponto de acesso de rede, ou **NAP** (**Network Access Point**). Essas operadoras eram PacBell (San Francisco), Ameritech (Chicago), MFS (Washington, D.C.) e Sprint (cidade de Nova York, onde, para fins de NAP, a localidade de Pennsauken, em Nova Jersey, pertence à cidade de Nova York). Todas as operadoras de redes que quisessem oferecer serviços de backbone às redes regionais da NSF tinham de estabelecer conexão com todos os NAPs.

Nessa estratégia, um pacote originário de uma das redes regionais tinha a opção de escolher uma das concessionárias de backbone para ser transferido do NAP de origem para o NAP de destino. Consequentemente, as concessionárias de backbone foram obrigadas a concorrer com as redes regionais, tendo de oferecer preços e serviços melhores para se manterem no mercado. Como resultado, o conceito de um único backbone padrão foi substituído por uma infraestrutura competitiva, com fins lucrativos. Muitas pessoas gostam de criticar o governo dos Estados Unidos por não ser inovador mas, na área de redes, foram o Departamento de Defesa e a NSF que criaram a infraestrutura que formou a base para a Internet, e depois a entregaram à indústria para cuidar de sua operação.

Durante a década de 1990, muitos outros países e regiões também construíram redes nacionais de pesquisa, geralmente moldadas de acordo com a ARPANET e a NSFNET. Na Europa, essas redes incluíram EuropaNET e EBONE, que começaram com linhas de 2 Mbps e depois foram atualizadas para linhas de 34 Mbps. Mais tarde, a infraestrutura de rede na Europa também foi entregue à indústria.

A Internet mudou muito dessa época para cá. Seu tamanho explodiu com o surgimento da World Wide Web (WWW), no início da década de 1990. Dados recentes do Internet Systems Consortium indicam que o número de hosts visíveis na Internet supera os 600 milhões. Esse número é apenas uma estimativa por baixo, mas ele é muito superior aos poucos milhões de hosts que existiam quando a primeira conferência sobre a WWW foi realizada no CERN, em 1994.

A maneira como usamos a Internet também mudou radicalmente. No início, aplicações como e-mail para aca-

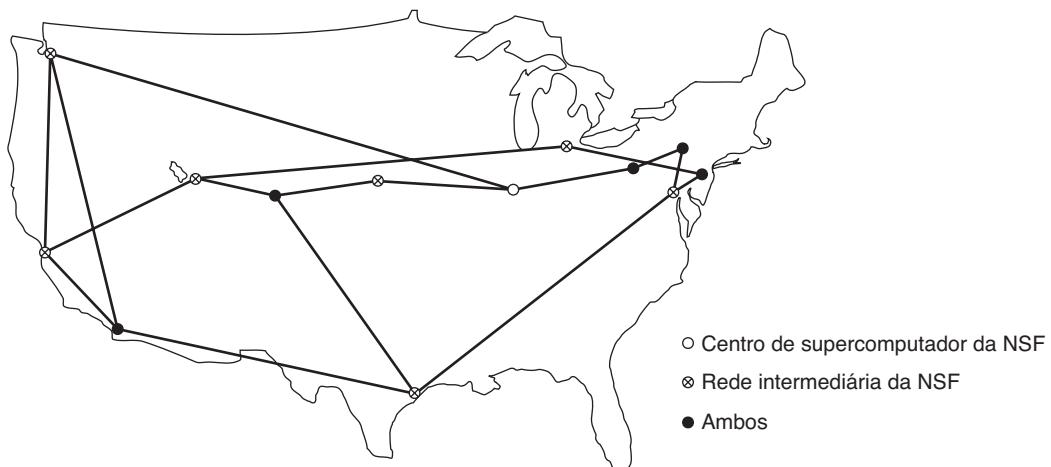



Figura 1.25 | O backbone da NSFNET em 1988.

dêmicos, grupos de notícias, login remoto e transferência de arquivos dominavam. Depois, ela passou a ser um e-mail para cada um, depois a Web e a distribuição de conteúdo peer-to-peer, como o Napster, hoje fora de operação. Atualmente, distribuição de mídia em tempo real, redes sociais (por exemplo, Facebook) e microblogging (por exemplo, Twitter) estão ganhando cada vez mais força. Essas mudanças trouxeram tipos de mídia mais ricos para a Internet e, consequentemente, muito mais tráfego. Na verdade, o tráfego dominante na Internet mudou com tanta regularidade que, por exemplo, novas e melhores maneiras de trabalhar com música ou filmes podem se tornar muito mais populares em pouco tempo.

Arquitetura da Internet

A arquitetura da Internet também mudou muito por ter crescido de forma explosiva. Nesta seção, tentaremos apresentar uma breve visão geral da Internet atual. O quadro é complicado pelas contínuas reviravoltas nos negócios das empresas telefônicas (telcos), empresas de cabo e ISPs, o que muitas vezes torna difícil saber quem está fazendo o quê. Um fator que impulsiona essas reviravoltas é a convergência das telecomunicações, em que uma rede é usada para fins anteriormente distintos. Por exemplo, em uma 'jogada tripla', uma empresa vende seu serviço de telefonia, TV e Internet na mesma conexão de rede, supondo que isso lhe fará economizar dinheiro. Consequentemente, essa descrição terá de ser um pouco mais simples que a realidade. E o que é verdade agora pode não ser verdade amanhã.

O quadro geral é mostrado na Figura 1.26. Agora, vamos examinar cada item dessa figura, começando com um computador doméstico (nas bordas do esquema). Para entrar na Internet, o computador é conectado a um **provedor de serviço de Internet**, ou **ISP** (**Internet Service Provider**), de quem o usuário compra **acesso à Internet** ou **conectividade**. Com isso, o computador pode trocar pacotes com todos os outros hosts acessíveis na Internet. O usuário pode enviar pacotes para navegar pela Web ou para qualquer um dos milhares de outros usos; isso não importa. Existem muitos tipos diferentes de acesso à Internet, e eles normalmente são distinguidos por quanta largura de banda oferecem e quanto custam, mas o atributo mais importante é a conectividade.

Um modo comum de se conectar a um ISP é usando a linha telefônica em sua casa, e, nesse caso, sua companhia telefônica é o seu ISP. A **DSL** (**Digital Subscriber Line**) reutiliza a linha telefônica que se conecta à sua casa para a transmissão de dados digitais. O computador é conectado a um dispositivo chamado **modem DSL**, que faz a conversão entre pacotes digitais e sinais analógicos que podem passar sem ser impedidos pela linha telefônica. Na outra ponta, um dispositivo chamado **DSLAM** (**Digital Subscriber Line Access Multiplexer**) faz a conversão entre sinais e pacotes.

A Figura 1.26 também mostra várias outras maneiras populares de se conectar a um ISP. A DSL é um modo de usar a linha telefônica local com maior largura de banda do que enviar bits por uma ligação telefônica tradicional em vez de uma conversa de voz. Isso é chamado **conexão discada** (**dial-up**) e é feito com um tipo diferente de modem nas duas extremidades. A palavra **modem** é uma contração de '*mo*dulador-*dem*odulador' e refere-se a qualquer dispositivo que faz a conversão entre bits digitais e sinais analógicos.

Outro método é enviar sinais pelo sistema de TV a cabo. Assim como a DSL, essa é uma maneira de reutilizar a infraestrutura existente — nesse caso, canais de TV a cabo não utilizados. O dispositivo na residência é chamado **modem a cabo**, e o dispositivo no **terminal de cabo** é chamado **CMTS** (**Cable Modem Termination System**).

DSL e cabo oferecem acesso à Internet em velocidades que variam desde uma pequena fração de um megabit/s

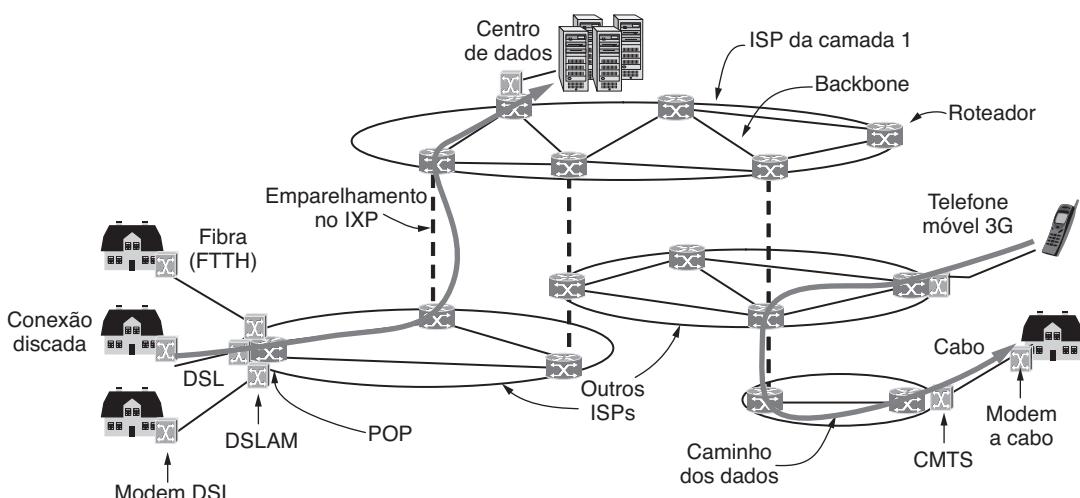



Figura 1.26 | Visão geral da arquitetura da Internet.

até vários megabits/s, dependendo do sistema. Essas velocidades são muito maiores que as da linha discada, que são limitadas a 56 kbps, em razão da estreita largura de banda usada para ligações de voz. O acesso à Internet em velocidades muito maiores que as discadas é chamado de **banda larga** (**broadband**). O nome refere-se à maior largura de banda usada para redes mais velozes, e não a qualquer velocidade em particular.

Os métodos de acesso mencionados até aqui são limitados pela largura de banda da 'última milha', ou última perna de transmissão. Levando a fibra óptica até as residências, o acesso mais rápido à Internet pode ser fornecido em velocidades na ordem de 10 a 100 Mbps. Esse esquema é conhecido como **FTTH** (**Fiber to the Home**). Para empresas em áreas comerciais, pode fazer sentido alugar uma linha de transmissão de alta velocidade dos escritórios até o ISP mais próximo. Por exemplo, na América do Norte, uma linha T3 trabalha a aproximadamente 45 Mbps.

O wireless também é usado para acesso à Internet. Um exemplo que mostraremos rapidamente é o das redes de telefonia móvel 3G. Elas podem oferecer entrega de dados em velocidades de 1 Mbps ou mais para telefones móveis e assinantes fixos na área de cobertura.

Agora, podemos mover pacotes entre a residência e o ISP. Chamamos o local em que os pacotes do cliente entram na rede do ISP de **ponto de presença**, ou **POP** (**Point of Presence**) do ISP. Em seguida, explicaremos como os pacotes são movimentados entre os POPs de diferentes ISPs. Desse ponto em diante, o sistema é totalmente digital e comutado por pacotes.

As redes do ISP podem ter escopo regional, nacional ou internacional. Já vimos que sua arquitetura é composta de linhas de transmissão de longa distância que interconectam roteadores nos POPs nas diferentes cidades que os ISPs atendem. Esse equipamento é chamado de **backbone** do ISP. Se um pacote é destinado a um host servido diretamente pelo ISP, esse pacote é roteado pelo backbone e entregue ao host. Caso contrário, ele deve ser entregue a outro ISP.

Os ISPs conectam suas redes para trocar tráfego nos **IXPs** (**Internet eXchange Points**). Diz-se que os ISPs conectados são **emparelhados** (**peer**). Existem muitos IXPs em cidades do mundo inteiro. Eles são desenhados verticalmente na Figura 1.26, pois as redes de ISP se sobrepõem geograficamente. Basicamente, um IXP é uma sala cheia de roteadores, pelo menos um por ISP. Uma LAN na sala conecta todos os roteadores, de modo que os pacotes podem ser encaminhados de qualquer backbone ISP para qualquer outro backbone ISP. Os IXPs podem ser instalações grandes e independentes. Um dos maiores é o Amsterdam Internet Exchange, ao qual se conectam centenas de ISPs e através do qual eles trocam centenas de gigabits/s de tráfego.

O emparelhamento que ocorre nos IXPs depende dos relacionamentos comerciais entre os ISPs. Existem muitos relacionamentos possíveis. Por exemplo, um ISP pequeno poderia pagar a um ISP maior pela conectividade à Internet para alcançar hosts distantes, assim como um cliente compra o serviço de um provedor de Internet. Nesse caso, diz-se que o ISP pequeno paga pelo **tráfego**. Como alternativa, dois ISPs grandes podem decidir trocar tráfego de modo que cada ISP possa entregar algum tráfego ao outro sem pagar pelo trânsito. Um dos muitos paradoxos da Internet é que os ISPs que concorrem por clientes publicamente uns com os outros normalmente trabalham em cooperação para realizar o emparelhamento (Metz, 2001).

O caminho que um pacote segue pela Internet depende das escolhas de emparelhamento dos ISPs. Se o ISP que entrega um pacote se emparelhar com o ISP de destino, ele pode entregar o pacote diretamente a seu par. Caso contrário, ele pode rotear o pacote para o local mais próximo em que se conecta a um provedor de trânsito pago, de modo que o provedor possa entregar o pacote. Dois exemplos de caminhos pelos ISPs são desenhados na Figura 1.26. Normalmente, o caminho seguido por um pacote não será o caminho mais curto pela Internet.

No topo dessa cadeia estão algumas operadoras importantes, como AT&T e Sprint, que operam grandes redes internacionais de backbones, com milhares de roteadores conectados por enlaces de fibra óptica de alta largura de banda. Esses ISPs não pagam pelo trânsito. Eles normalmente são chamados ISPs da **camada 1** e formam o backbone principal da Internet, pois todos os outros devem se conectar a eles para poderem alcançar a Internet inteira.

Empresas que fornecem muito conteúdo, como Google e Yahoo!, localizam seus computadores em **centros de dados** que estão bem conectados com o restante da Internet. Esses centros de dados são projetados para computadores, não para humanos, e podem estar cheios de racks e mais racks de máquinas, o que chamamos de **parque de servidores**. A **colocalização** ou a **hospedagem** de centros de dados permite que os clientes coloquem equipamentos como servidores nos POPs do ISP, de modo que possa haver conexões curtas e rápidas entre os servidores e os backbones do ISP. O setor de hospedagem da Internet tornou-se cada vez mais virtualizado, de modo que agora é comum alugar uma máquina virtual, executada em um parque de servidores, em vez de instalar um computador físico. Esses centros de dados são tão grandes (dezenas ou centenas de milhares de máquinas) que a eletricidade tem um grande custo, de modo que esses centros às vezes são construídos em locais onde a eletricidade é mais barata.

Isso termina nosso rápido passeio pela Internet. Nos próximos capítulos, falaremos muito mais sobre os componentes individuais e seus projetos, algoritmos e protocolos. Outro ponto que merece ser mencionado aqui é que o conceito de estar na Internet está mudando. Antigamente, uma máquina estava na Internet se ela: (1) executasse a pilha de protocolos TCP/IP; (2) tivesse um endereço IP;

e (3) pudesse enviar pacotes IP a todas as outras máquinas na Internet. Contudo, os ISPs normalmente reutilizam endereços IP, dependendo de quais computadores estão em uso no momento, e as redes domésticas normalmente compartilham um endereço IP entre vários computadores. Essa prática anula a segunda condição. Medidas de segurança, como firewalls, também podem impedir parcialmente que computadores recebam pacotes, anulando a terceira condição. Apesar dessas dificuldades, faz sentido considerar tais máquinas como estando na Internet, embora estejam conectadas a seus ISPs.

Também devemos mencionar, de passagem, que algumas empresas têm interligado todas as suas redes internas existentes, normalmente usando a mesma tecnologia da Internet. Essas **intranets** normalmente são acessíveis apenas nas instalações da empresa ou a partir de notebooks da empresa, mas seu funcionamento é idêntico ao da Internet.

1.5.2 Redes de telefonia móvel de terceira geração

As pessoas adoram falar ao telefone ainda mais do que navegar pela Internet, e isso tem tornado a rede de telefonia móvel a rede mais bem-sucedida do mundo. Ela tem mais de quatro bilhões de assinantes no mundo inteiro. Para entender melhor esse número, ele significa aproximadamente 60 por cento da população do mundo, e é superior ao número de hosts da Internet e linhas telefônicas fixas juntos (ITU, 2009).

A arquitetura da rede de telefonia móvel mudou bastante durante os últimos quarenta anos, junto com seu incrível crescimento. Os sistemas de telefonia móvel de primeira geração transmitiam chamadas de voz como sinais com variação contínua (analógicos) ao invés de sequências de bits (digitais). O **sistema avançado de telefonia móvel**, ou **AMPS** (**Advanced Mobile Phone System**), implantado nos Estados Unidos em 1982, era um sistema de primeira geração bastante utilizado. Os sistemas de telefonia móvel de segunda geração comutavam para transmitir chamadas de voz em forma digital para aumentar a capacidade, melhorar a segurança e oferecer mensagens de texto. Um exemplo de sistema 2G é o **sistema global para comunicações móveis**, ou **GSM** (**Global System for Mobile communications**), que foi implantado a partir de 1991 e tornou-se o sistema de telefonia móvel mais usado no mundo.

Os sistemas de terceira geração, ou 3G, foram implantados inicialmente em 2001 e oferecem tanto voz digital como serviços de dados digitais de banda larga. Eles também vêm com muito jargão e muitos padrões diferentes à sua escolha. O 3G é definido de forma livre pela ITU (uma agência de padrões internacionais que discutiremos na próxima seção) como fornecendo velocidades de pelo menos 2 Mbps para usuários parados ou caminhando, e 384 kbps em um veículo em movimento. O **UMTS** (**Universal Mobile Telecommunications System**), também chamado **WCDMA** (**Wideband Code Division Multiple Access**), é o principal sistema 3G que está sendo rapidamente implantado no mundo inteiro. Ele pode oferecer até 14 Mbps no downlink (enlace de descida) e quase 6 Mbps no uplink (enlace de subida). Versões futuras usarão antenas múltiplas e rádios para fornecer aos usuários velocidades ainda maiores.

O recurso escasso nos sistemas 3G, assim como nos sistemas 2G e 1G antes deles, é o espectro de radiofrequência. Os governos licenciam o direito de usar partes do espectro para as operadoras de rede de telefonia móvel, em geral usando um leilão de espectro em que as operadoras de rede submetem ofertas. Ter uma parte do espectro licenciado facilita o projeto e a operação dos sistemas, uma vez que ninguém mais tem permissão para transmitir nesse espectro, mas isso normalmente custa muito dinheiro. No Reino Unido em 2000, por exemplo, cinco licenças 3G foram leiloadas por um total de cerca de US$ 40 bilhões.

É a escassez do espectro que ocasionou o projeto de **rede celular** mostrado na Figura 1.27, que agora é usado para redes de telefonia móvel. Para controlar a interferência de rádio entre os usuários, a área de cobertura é dividida em células. Dentro de uma célula, os usuários recebem canais que não interferem uns com os outros e não causam muita interferência nas células adjacentes. Isso permite uma boa reutilização do espectro, ou **reutilização de frequência**, nas células vizinhas, o que aumenta a capacidade da rede. Nos sistemas 1G, que transportavam cada canal de voz em uma banda de frequência específica, as frequências eram cuidadosamente escolhidas, de modo que não entrassem em conflito com as células vizinhas. Desse modo, uma dada frequência só poderia ser reutilizada uma vez em várias células. Os sistemas 3G modernos permitem que cada célula utilize todas as frequências, mas de um modo que resulte em um nível tolerável de interferência com as células vizinhas. Existem variações no projeto celular, incluindo o uso de antenas direcionais ou setorizadas nas torres das células, para reduzir ainda mais a interferência, mas a ideia básica é a mesma.

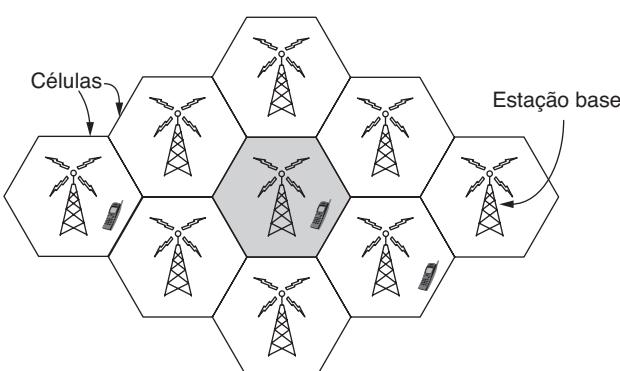



Figura 1.27 | Projeto celular de redes de telefonia móvel.

A arquitetura da rede de telefonia móvel é muito diferente da arquitetura da Internet. Ela tem várias partes, como vemos na versão simplificada da arquitetura UMTS da Figura 1.28. Primeiro, existe uma **interface com o ar.** Esse termo é um nome elegante para o protocolo de comunicação por rádio, usado pelo ar entre os dispositivos móveis (por exemplo, o telefone celular) e a **estação-base celular**. Os avanços na interface com o ar durante as últimas décadas aumentaram bastante as velocidades dos dados sem fios. A interface com o ar no UMTS é baseada em **Code Division Multiple Access** (**CDMA**), uma técnica que estudaremos no Capítulo 2.

A estação-base da rede celular forma, com seu controlador, a **rede de acesso por rádio**. Essa parte é o lado sem fios da rede de telefonia móvel. O nó controlador ou **RNC** (**Radio Network Controller**) controla como o espectro é utilizado. A estação-base implementa a interface com o ar. Ela é chamada **Nó B**, um rótulo temporário que pegou.

O restante da rede de telefonia móvel transporta o tráfego para a rede de acesso por rádio. Ela é chamada **núcleo da rede**. O núcleo da rede UMTS evoluiu a partir do núcleo da rede usada para o sistema 2G GSM que veio antes dela. Contudo, algo surpreendente está acontecendo no núcleo da rede UMTS.

Desde o início das redes, uma guerra estava sendo travada entre as pessoas que apoiam redes de pacotes (ou seja, sub-redes não orientadas a conexões) e as pessoas que apoiam redes de circuitos (ou seja, sub-redes orientadas a conexões). Os principais componentes dos pacotes vêm da comunidade da Internet. Em um projeto não orientado a conexões, cada pacote é roteado independentemente um do outro. Por conseguinte, se alguns roteadores deixarem de funcionar durante uma sessão de comunicação, nenhum dano será feito desde que o sistema possa ser reconfigurado dinamicamente, de modo que os pacotes subsequentes possam encontrar alguma rota até o destino, mesmo que seja diferente daquele que os pacotes anteriores usaram.

O circuito de campo vem do mundo das companhias telefônicas. No sistema telefônico convencional, uma pessoa precisa digitar um número e esperar uma conexão antes de falar ou enviar dados. Esse esquema de conexão estabelece uma rota através do sistema telefônico que é mantida até que a chamada termine. Todas as palavras ou pacotes seguem a mesma rota. Se uma linha ou uma central no caminho for interrompida, a chamada é cancelada, tornando-a menos tolerante a falhas do que um projeto não orientado a conexões.

A vantagem dos circuitos é que eles podem dar suporte mais fácil à qualidade de serviço. Estabelecendo uma conexão com antecedência, a sub-rede pode reservar recursos como largura de banda do enlace, melhor espaço de buffer e de CPU do switch. Se alguém tentar estabelecer uma chamada e não houver recursos suficientes, a chamada é rejeitada e quem liga recebe um sinal de ocupado. Desse modo, quando uma conexão é estabelecida, a conexão receberá um serviço bom.

Com uma rede não orientada a conexões, se muitos pacotes chegarem ao mesmo roteador no mesmo instante, o roteador ficará sobrecarregado e provavelmente perderá pacotes. Por fim, o emissor notará isso e os enviará novamente, mas a qualidade do serviço será fraca e inadequada para áudio ou vídeo, a menos que a rede esteja pouco carregada. Nem é preciso dizer que fornecer uma qualidade de áudio adequada é algo com que as companhias telefônicas se importam muito, daí sua preferência por conexões.

A surpresa na Figura 1.28 é que existe equipamento de comutação de pacotes nos circuitos do núcleo da rede. Isso mostra a rede de telefonia móvel em transição, com as companhias de telefonia móvel capazes de implemen-

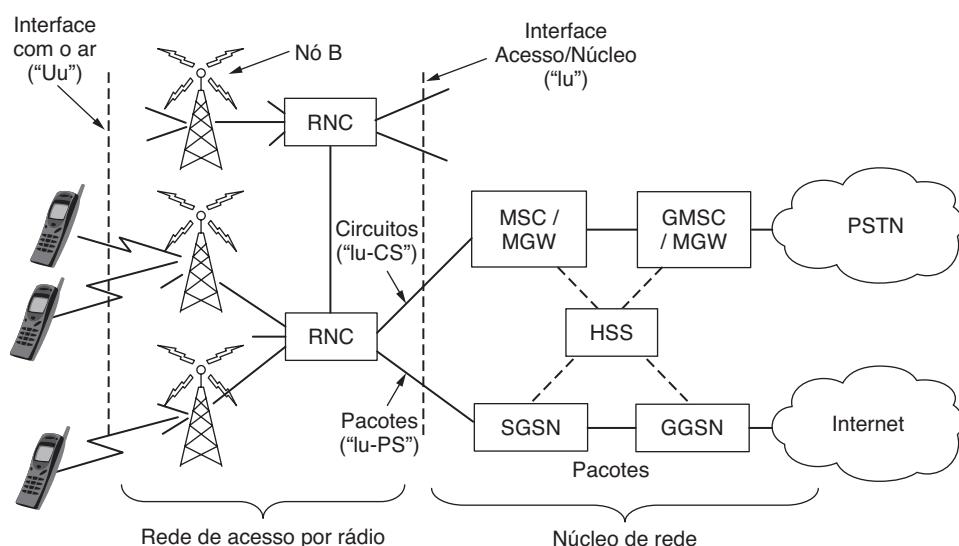



Figura 1.28 | Arquitetura da rede de telefonia móvel 3G UMTS.

tar uma ou, às vezes, ambas as alternativas. As redes de telefonia móvel mais antigas usavam um núcleo comutado por circuitos no estilo da rede telefônica tradicional para transportar chamadas de voz. Esse legado é visto na rede UMTS com os elementos **MSC** (**Mobile Switching Center**), **GMSC** (**Gateway Mobile Switching Center**) e **MGW** (**Media Gateway**), que estabelecem conexões por um núcleo de rede com comutação de circuitos, como a **PSTN** (**Public Switched Telephone Network**).

Os serviços de dados se tornaram uma parte muito mais importante da rede de telefonia móvel do que costumavam ser, começando com mensagens de texto e os primeiros serviços de dados por pacotes, como **GPRS** (**General Packet Radio Service**) no sistema GSM. Esses serviços de dados mais antigos funcionavam em dezenas de kbps, mas os usuários queriam mais. As redes de telefonia móvel mais novas transportam pacotes de dados em velocidades múltiplas de Mbps. Para comparação, uma chamada de voz é feita a 64 kbps, normalmente três a quatro vezes menos com compactação.

Para transportar todos esses dados, os nós do núcleo da rede UMTS se conectam diretamente a uma rede de comutação de pacotes. O **SGSN** (**Serving GPRS Support Node**) e o **GGSN** (**Gateway GPRS Support Node**) entregam pacotes de dados de e para telefones móveis e se conectam a redes externas de pacotes, como a Internet.

Essa transição deverá continuar nas redes de telefonia móvel que estão sendo planejadas e implementadas agora. Os protocolos da Internet ainda são utilizados em telefones móveis para estabelecer conexões para chamadas de voz por uma rede de dados de pacotes, na forma de VoIP (Voice over IP). IP e pacotes são usados desde o acesso via rádio até o acesso ao núcleo da rede. Naturalmente, o modo como as redes IP são projetadas também está mudando para dar melhor suporte à qualidade do serviço. Se isso não ocorrer, problemas com áudio e vídeo picotados não impressionarão os clientes pagantes. Retornaremos a esse assunto no Capítulo 5.

Outra diferença entre redes de telefonia móvel e a Internet tradicional é a mobilidade. Quando um usuário sai do alcance de uma estação-base celular e entra no alcance de outra, o fluxo de dados deve ser redirecionado da estação-base antiga para a nova. Essa técnica é conhecida como **transferência** (**handover** ou **handoff**), como ilustra a Figura 1.29.

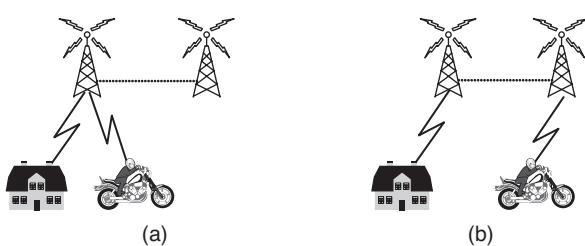



Figura 1.29 | Handover de telefone móvel (a) antes, (b) depois.

Ou o dispositivo móvel ou a estação-base podem solicitar uma transferência quando a qualidade do sinal cai. Em algumas redes de celular, normalmente nas baseadas na tecnologia CDMA, é possível se conectar à nova estação-base antes de se desconectar da estação antiga. Isso melhora a qualidade da conexão para o telefone móvel, pois não existe interrupção no serviço; o aparelho na verdade está conectado às duas estações-base por um pequeno período. Esse modo de fazer uma transferência é chamado de **soft handover**, para distingui-lo do **hard handover**, em que o aparelho se desconecta da estação-base antiga antes de se conectar à nova.

Uma questão relacionada é como encontrar um aparelho móvel em primeiro lugar quando existe uma chamada. Cada rede de telefonia móvel tem um **HSS** (**Home Subscriber Server**) no núcleo da rede, que sabe o local de cada assinante, bem como outras informações de perfil usadas para autenticação e autorização. Desse modo, cada aparelho poderá ser encontrado contatando o HSS.

Uma última área que deve ser discutida é a segurança. Historicamente, as companhias telefônicas têm levado a segurança muito mais a sério do que as empresas da Internet há muito tempo, em razão da necessidade de cobrar pelo serviço e evitar fraudes (no pagamento). Infelizmente, isso não diz muita coisa. Apesar disso, na evolução das tecnologias entre 1G e 3G, as companhias de telefonia móvel conseguiram implementar alguns mecanismos de segurança básicos para os aparelhos móveis.

A partir do sistema 2G GSM, o telefone móvel foi dividido em um aparelho e um chip removível, contendo as informações de identidade e conta do assinante. O chip é chamado informalmente de **cartão SIM**, abreviação de **Subscriber Identity Module** (módulo de identificação do assinante). Os cartões SIM podem ser trocados para aparelhos diferentes para serem ativados, e oferecem uma base para a segurança. Quando os clientes GSM viajam para outros países, em férias ou a negócios, eles normalmente levam seus aparelhos, mas compram um novo cartão SIM por alguns dólares quando chegam lá, a fim de fazer ligações locais sem pagar pelo roaming.

Para diminuir fraudes, as informações nos cartões SIM também são usadas pela rede de telefonia móvel para autenticar os assinantes e verificar se eles têm permissão para usar a rede. Com UMTS, o aparelho móvel também usa as informações no cartão SIM para verificar se está falando com uma rede legítima.

Outro aspecto da segurança é a privacidade. Os sinais sem fio são transmitidos para todos os receptores vizinhos, de modo que, para dificultar a escuta das conversas, chaves criptográficas no cartão SIM são usadas para encriptar as transmissões. Essa técnica oferece uma privacidade muito melhor do que os sistemas 1G, que eram facilmente interceptados, mas não resolve todos os problemas, em virtude das deficiências nos esquemas de encriptação.

As redes de telefone móvel se destinam a desempenhar um papel central nas redes do futuro. Elas agora são mais aplicações de banda larga móvel do que chamadas de voz, e isso tem grandes implicações nas interfaces com o ar, a arquitetura do núcleo da rede e a segurança das redes do futuro. Tecnologias 4G, mais rápidas e melhores, estão em desenvolvimento sob o nome de **LTE** (**Long Term Evolution**), mesmo enquanto o projeto e a implementação do 3G continuam. Outras tecnologias sem fios também oferecem acesso à Internet de banda larga para clientes fixos e móveis, em especial as redes 802.16, sob o nome comum de **WiMAX**. É totalmente possível que LTE e WiMAX estejam em curso de colisão, e é difícil prever o que acontecerá com elas.

1.5.3 LANs sem fios: 802.11

Quase na mesma época em que surgiram os notebooks, muitas pessoas sonhavam com o dia em que entrariam em um escritório e, como mágica, seus notebooks se conectariam à Internet. Em consequência disso, diversos grupos começaram a trabalhar para descobrir maneiras de alcançar esse objetivo. A abordagem mais prática é equipar o escritório e os notebooks com transmissores e receptores de rádio de ondas curtas para permitir a comunicação entre eles.

Esse trabalho rapidamente levou à comercialização de LANs sem fios por várias empresas. O problema era encontrar duas delas que fossem compatíveis. Essa proliferação de padrões significava que um computador equipado com um rádio da marca *X* não funcionaria em uma sala equipada com uma estação-base da marca *Y*. Em meados da década de 1990, a indústria decidiu que um padrão de LAN sem fios poderia ser uma boa ideia, e assim o comitê do IEEE que padronizou as LANs com fios recebeu a tarefa de elaborar um padrão de LANs sem fios.

A primeira decisão foi a mais fácil: como denominá-lo. Todos os outros padrões de LANs tinham números como 802.1, 802.2, 802.3, até 802.10; assim, o padrão de LAN sem fio recebeu a denominação 802.11. Uma gíria para ele é **WiFi**, mas esse é um padrão importante e merece respeito, de modo que o chamaremos pelo seu nome correto, 802.11.

O restante era mais difícil. O primeiro problema foi descobrir uma banda de frequência adequada que estivesse disponível, de preferência em todo o mundo. A técnica empregada foi o oposto da que nas redes de telefonia móvel. Ao invés do caro espectro licenciado, os sistemas 802.11 operam nas bandas não licenciadas, como as bandas **ISM** (**Industrial, Scientific, and Medical**) definidas pela ITU-R (por exemplo, 902-928 MHz, 2,4-2,5 GHz, 5,725-5,825 GHz). Todos os dispositivos têm permissão para usar esse espectro, desde que limitem sua potência de transmissão para permitir que diferentes dispositivos coexistam. Naturalmente, isso significa que os rádios 802.11 podem estar competindo com telefones sem fio, aparelhos para abrir portas de garagem e fornos de micro-ondas.

As redes 802.11 são compostas de clientes, como notebooks e telefones móveis, e infraestrutura chamada **pontos de acesso**, ou **APs** (**Access Points**), que são instalados nos prédios. Os pontos de acesso também são chamados de **estações-base**. Os pontos de acesso se conectam à rede com fios, e toda a comunicação entre os clientes passa por um ponto de acesso. Também é possível que os clientes no alcance do rádio falem diretamente, como dois computadores em um escritório sem um ponto de acesso. Esse arranjo é chamado de **rede ocasional** (ou **rede ad hoc**). Ele é usado com muito menos frequência do que o modo de ponto de acesso. Os dois modos aparecem na Figura 1.30.

A transmissão 802.11 é complicada pelas condições da rede sem fio, que variam até mesmo com pequenas mudanças no ambiente. Nas frequências usadas para 802.11, os sinais de rádio podem ser refletidos por objetos sólidos, de modo que vários ecos de uma transmissão podem alcançar um receptor por diferentes caminhos. Os ecos podem cancelar ou reforçar um ao outro, fazendo com que o sinal recebido flutue bastante. Esse fenômeno é chamado **enfraquecimento por múltiplos caminhos** (ou **multipath fading**), e pode ser visto na Figura 1.31.

A ideia-chave para contornar as condições variáveis da rede sem fios é a **diversidade de caminhos**, ou o envio de informações por vários caminhos independentes. Desse modo, a informação provavelmente será recebida mesmo

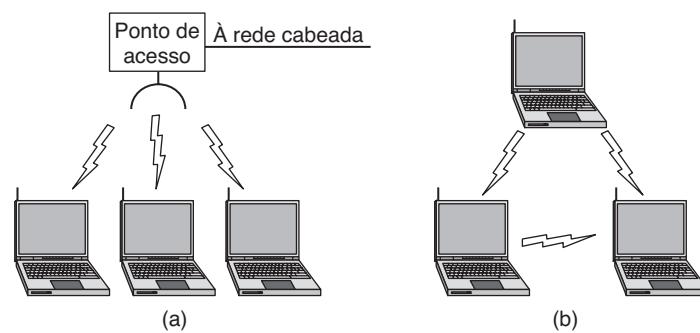



Figura 1.30 | (a) Rede sem fio com um ponto de acesso. (b) Rede ocasional.

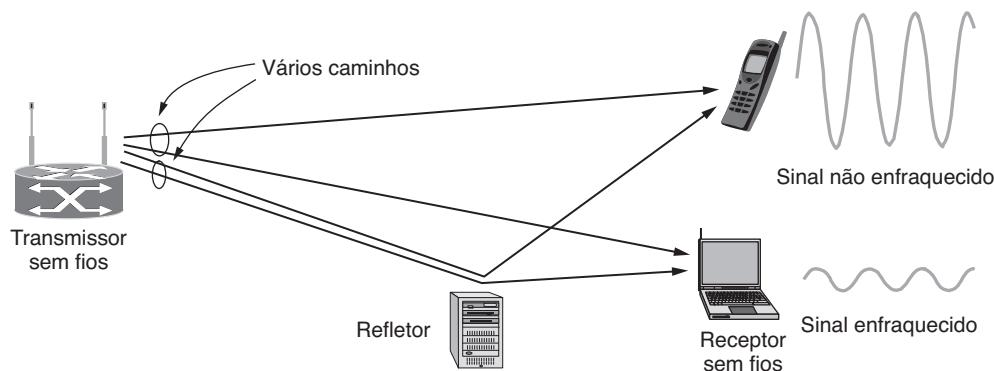



Figura 1.31 | Enfraquecimento por múltiplos caminhos.

que um dos caminhos seja ruim, em decorrência de um enfraquecimento. Esses caminhos independentes normalmente são embutidos no esquema de modulação digital, na camada física. As opções incluem usar diferentes frequências pela banda permitida, seguir diferentes caminhos espaciais entre diferentes pares de antenas ou repetir os bits por diferentes períodos.

Diferentes versões do 802.11 têm usado todas essas técnicas. O padrão inicial (de 1997) definia uma LAN sem fios que funcionava a 1 ou 2 Mbps pelo salto entre frequências ou espalhamento do sinal pelo espectro permitido. Quase imediatamente, as pessoas reclamaram que ela era muito lenta, de modo que o trabalho foi iniciado com padrões mais rápidos. O projeto do espectro espalhado foi estendido e tornou-se o padrão 802.11b (1999), funcionando em velocidades de até 11 Mbps. Os padrões 802.11a (1999) e 802.11g (2003) passaram para um esquema de modulação diferente, chamado **multiplexação ortogonal por divisão de frequência**, ou **OFDM** (**Orthogonal Frequency Division Multiplexing**). Este divide uma banda larga do espectro em muitas fatias estreitas, sobre as quais diferentes bits são enviados em paralelo. Esse esquema melhorado, que estudaremos no Capítulo 2, aumentou a velocidade do 802.11a/g para 54 Mbps. Esse é um aumento significativo, mas as pessoas ainda queriam mais vazão para dar suporte a usos mais exigentes. A versão mais recente é o 802.11n (2009). Ela usa bandas de frequência maiores e até quatro antenas por computador, para alcançar velocidades de até 450 Mbps.

Como o meio sem fio é inerentemente de difusão, os rádios 802.11 também precisam lidar com o problema de que múltiplas transmissões enviadas ao mesmo tempo colidirão, o que pode interferir na recepção. Para lidar com esse problema, o 802.11 usa um **esquema de acesso múltiplo com detecção de portador**, **CSMA** (**Carrier Sense Multiple Access**), com base nas ideias da rede Ethernet clássica com fios, que, ironicamente, se baseava em uma antiga rede sem fio desenvolvida no Havaí, chamada **ALOHA**. Os computadores esperam por um intervalo aleatório antes de transmitir, e adiam suas transmissões se descobrirem que mais alguém já está transmitindo. Esse esquema torna menos provável que dois computadores enviem ao mesmo tempo. Porém, ele não funciona tão bem quanto no caso das redes com fios. Para entender por que, examine a Figura 1.32. Suponha que o computador *A* esteja transmitindo para o computador *B*, mas o alcance de rádio do transmissor de *A* é muito curto para alcançar o computador *C*. Se *C* quiser transmitir para *B*, ele pode escutar antes de começar, mas o fato de ele não escutar nada não significa que sua transmissão terá sucesso. A impossibilidade de *C* escutar *A* antes de começar faz com que ocorram colisões. Depois de qualquer colisão, o emissor então espera outro período aleatório maior e retransmite o pacote. Apesar disso e de algumas outras questões, o esquema funciona muito bem na prática.

Outro problema é a mobilidade. Se um cliente móvel se afastar do ponto de acesso que está usando e seguir para o alcance de um ponto de acesso diferente, é preciso haver alguma forma de transferência (handover). A solução é que uma rede 802.11 pode consistir em múltiplas células, cada uma com seu próprio ponto de acesso, e um sistema de distribuição que conecta essas células. O sistema de distribuição normalmente é a Ethernet comutada, mas ele pode usar qualquer tecnologia. À medida que os clientes

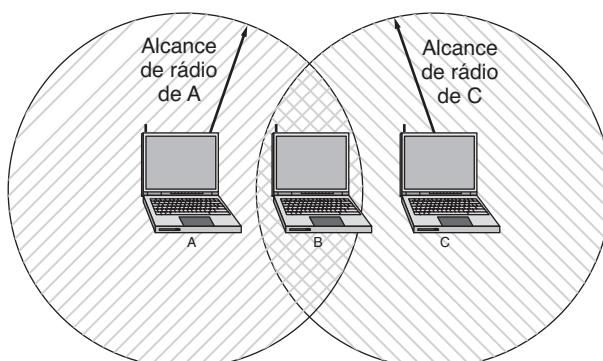



Figura 1.32 | O alcance de um único rádio pode não abranger o sistema inteiro.

se movem, eles podem encontrar outro ponto de acesso com um sinal melhor que o usado atualmente e mudar sua associação. De fora, o sistema inteiro se parece com uma única LAN com fios.

Assim, a mobilidade no 802.11 tem sido limitada em comparação com a mobilidade na rede de telefonia móvel. Normalmente, o 802.11 é usado por clientes que vão de um local fixo para outro, ao invés de ser usado em trânsito. A mobilidade não é realmente necessária para uso dessa forma. Até mesmo quando a mobilidade do 802.11 é usada, ela se estende por uma única rede 802.11, que poderia cobrir, no máximo, um prédio grande. Futuros esquemas precisarão oferecer mobilidade por diferentes redes e diferentes tecnologias (por exemplo, 802.21).

Finalmente, existe o problema da segurança. Como as transmissões sem fio são feitas por radiodifusão, é fácil que computadores vizinhos recebam pacotes de informações que não foram solicitados por eles. Para evitar isso, o padrão 802.11 incluiu um esquema de encriptação conhecido como **WEP** (**Wired Equivalent Privacy**). A ideia foi tornar a segurança da rede sem fios semelhante à segurança da rede cabeada. Essa é uma boa ideia, mas infelizmente o esquema tinha falhas e logo foi quebrado (Borisov et al., 2001). Desde então, ele foi substituído por esquemas mais recentes, que possuem diferentes detalhes criptográficos no padrão 802.11i, também chamado **WiFi Protected Access**, inicialmente **WPA** e depois substituído pelo **WPA2**.

O 802.11 tem causado uma revolução nas redes sem fios, que ainda deverá continuar. Além de prédios, ele está começando a ser instalado em trens, aviões, barcos e automóveis, de modo que as pessoas possam navegar pela Internet enquanto viajam. Os telefones móveis e todo os tipos de aparelhos de consumo, de consoles de jogos a câmeras digitais, podem se comunicar com ele. Voltaremos a esse assunto com mais detalhes no Capítulo 4.

1.5.4 RFID e redes de sensores

As redes que estudamos até aqui são compostas de dispositivos de comunicação fáceis de reconhecer, de computadores a telefones móveis. Com a **identificação por radiofrequência**, ou **RFID** (**Radio Frequency IDentification**), os objetos do cotidiano também podem fazer parte de uma rede de computadores.

Uma etiqueta RFID se parece com um selo postal adesivo, que pode ser afixado (ou embutido) em um objeto para que possa ser rastreado. O 'objeto' pode ser uma vaca, um passaporte, um livro ou um palete de armazenagem. A etiqueta consiste em um pequeno microchip com um identificador exclusivo e uma antena que recebe transmissões de rádio. Leitores de RFID instalados nos pontos de rastreamento encontram as etiquetas quando elas entram no alcance e solicitam informações, como mostra a Figura 1.33. As aplicações incluem verificação de identidades, gerenciamento da cadeia de suprimentos, marcação de tempo em corridas e substituição de códigos de barras.

Existem muitos tipos de RFID, cada um com diferentes propriedades, mas talvez o aspecto mais fascinante dessa tecnologia seja que a maioria das tags RFID não possui uma tomada elétrica ou uma bateria. Ao invés disso, toda a energia necessária para operá-las é fornecida na forma de ondas de rádio pelos leitores de RFID. Essa tecnologia é chamada **RFID passiva**, para distingui-la da **RFID ativa** (menos comum), em que existe uma fonte de energia na etiqueta.

Uma forma comum de RFID é a **UHF RFID** (**Ultra-High Frequency RFID**). Ela é usada em paletes de armazenagem e em algumas carteiras de habilitação. Os leitores enviam sinais na banda de 902-928 MHz nos Estados Unidos. As etiquetas se comunicam em distâncias de vários metros, mudando o modo como elas refletem os sinais do leitor; este é capaz de apanhar essas reflexões. Esse modo de operação é chamado **refletor** (ou **backscatter**).

Outro tipo popular de RFID é o de alta frequência, ou **HF RFID** (**High Frequency RFID**). Ele opera a 13,56 MHz e provavelmente está em seus passaportes, cartões de crédito, livros e sistemas de pagamento sem contato. A HF RFID tem um alcance curto, normalmente de um metro ou menos, pois o mecanismo físico é baseado na indução, ao invés de no refletor. Também há outras formas de RFID usando outras frequências, como a **RFID de baixa frequência**, ou **LF RFID** (**Low Frequency RFID**), que foi desenvolvida antes da HF RFID e é usada para rastreamento de animais. Esse é o tipo de RFID que provavelmente estará em seu gato.

Leitores de RFID precisam solucionar de alguma forma o problema de lidar com várias etiquetas dentro do alcance de leitura. Isso significa que uma etiqueta não pode sim-

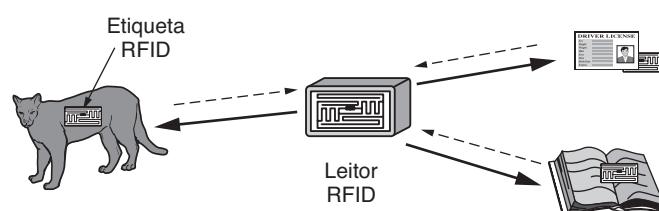



Figura 1.33 | RFID usada em objetos de rede do dia a dia.

plesmente responder quando escutar um leitor, ou então os sinais de várias etiquetas poderão colidir. A solução é semelhante à técnica usada no 802.11: as etiquetas esperam por um pequeno intervalo aleatório antes de responder com sua identificação, permitindo que o leitor focalize etiquetas individuais e as interrogue mais a fundo.

A segurança é outro problema. A capacidade dos leitores de RFID de rastrear um objeto com facilidade, e, portanto, a pessoa que o utiliza, pode ser uma invasão de privacidade. Infelizmente, é difícil proteger etiquetas RFID, pois elas não possuem o poder de computação e comunicação para executar algoritmos criptográficos fortes. Ao invés disso, são usadas medidas fracas, como senhas (que podem ser facilmente descobertas). Se uma carteira de identidade pode ser lida remotamente por um guarda em uma fronteira, o que impede que essa mesma carteira seja rastreada por outra pessoa sem seu conhecimento? Nada.

As etiquetas de RFID começaram como chips de identificação, mas estão rapidamente se transformando em computadores completos. Por exemplo, muitas etiquetas possuem memória, que pode ser atualizada e consultada mais tarde, de modo que a informação sobre o que aconteceu com o objeto etiquetado pode ser armazenada com ela. Rieback et al. (2006) demonstraram que isso significa que todos os problemas comuns de malware em computador se aplicam, portanto, agora seu gato ou seu passaporte poderiam ser usados para espalhar um vírus de RFID.

Um passo adiante na capacidade da RFID é a **rede de sensores**. Estas redes são implantadas para monitorar aspectos do mundo físico. Até agora, elas têm sido usadas principalmente para experimentos científicos, como o monitoramento de hábitat de pássaros, atividade vulcânica e migração de zebras, mas as aplicações comerciais, como cuidados de saúde, equipamento de monitoramento para vibrações e rastreamento de produtos congelados, refrigerados ou outros produtos perecíveis, não podem ser deixadas para trás.

Os nós de sensores são pequenos computadores, normalmente do tamanho de um token de segurança, que possuem sensores de temperatura, vibração e outros. Muitos nós são colocados no ambiente a ser monitorado. Normalmente, eles possuem baterias, embora possam recolher energia de vibrações ou do sol. Assim como a RFID, ter energia suficiente é um desafio importante, e os nós precisam se comunicar cuidadosamente para poderem entregar suas informações de sensoriamento a um ponto de coleta externo. Uma estratégia comum é que os nós se auto-organizem para repassar mensagens uns para os outros, como mostra a Figura 1.34. Esse projeto é chamado **rede de múltiplos saltos** (ou **rede multihop**).

A RFID e as redes de sensores provavelmente se tornarão muito mais capazes e difundidas no futuro. Os pesquisadores já combinaram o melhor das duas tecnologias, criando protótipos de etiquetas RFID programáveis, com sensores de luz, movimento e outros (Sample et al., 2008).

1.6 Padronização de redes

Existem muitos fabricantes e fornecedores de redes, cada qual com suas próprias ideias sobre como as coisas devem ser feitas. Sem coordenação, haveria um caos completo, e os usuários nada conseguiriam. A única opção de que a indústria dispõe é a criação de alguns padrões de rede. Além de permitirem que diferentes computadores se comuniquem, os padrões também ampliam o mercado para os produtos que aderem às suas regras. Um mercado mais amplo estimula a produção em massa, proporciona economia de escala no processo de produção, melhores implementações e outros benefícios que reduzem o preço e aumentam mais ainda a aceitação de um produto.

Nesta seção, examinaremos rapidamente o importante, mas pouco conhecido, mundo da padronização internacional. Porém, primeiro vamos discutir o que pertence a um padrão. Uma pessoa razoável poderia supor que um padrão lhe informe como um protocolo deverá funcionar, para que você possa fazer um bom trabalho de implementação. Essa pessoa estaria errada.

Os padrões definem o que é necessário para a interoperabilidade: nem mais, nem menos. Isso permite o surgimento de. um mercado maior e também que as empre-

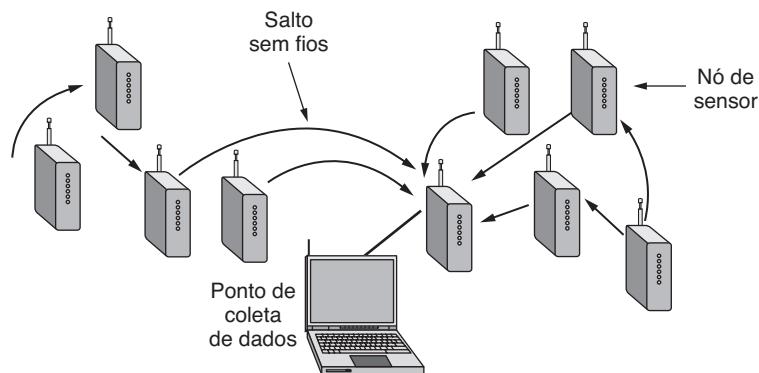



Figura 1.34 | Topologia multihop de uma rede de sensores.

sas disputem com base na qualidade de seus produtos. Por exemplo, o padrão 802.11 define muitas velocidades de transmissão, mas não diz quando um emissor deve usar qual velocidade, o que é um fator essencial no bom desempenho. Isso fica a critério de quem fabrica o produto. Geralmente, conseguir interoperabilidade dessa maneira é difícil, pois existem muitas escolhas de implementação e os padrões normalmente definem muitas opções. Para o 802.11, havia tantos problemas que, em uma estratégia que se tornou uma prática comum, um grupo comercial chamado **WiFi Alliance** foi iniciado para trabalhar com a interoperabilidade dentro do padrão 802.11.

De modo semelhante, um padrão de protocolo define o protocolo usado, mas não a interface de serviço internamente, exceto para ajudar a explicar o próprio protocolo. As interfaces de serviço reais normalmente são patenteadas. Por exemplo, não importa o modo como o TCP realiza interface com o IP dentro de um computador para falar com um host remoto. Só importa que o host remoto fale TCP/IP. De fato, TCP e IP normalmente são implementados juntos, sem qualquer interface distinta. Com isso, boas interfaces de serviço, assim como boas APIs, são valiosas por usar os protocolos, e as melhores (como os soquetes de Berkeley) podem se tornar muito populares.

Os padrões se dividem em duas categorias: de fato e de direito. Os padrões **de fato** são aqueles que se consagraram naturalmente, sem nenhum plano formal. O HTTP, protocolo no qual a Web funciona, começou como um padrão de fato. Ele fazia parte dos primeiros navegadores da WWW desenvolvidos por Tim Berners-Lee no CERN, e seu uso decolou com o crescimento da Web. O Bluetooth é outro exemplo. Ele foi desenvolvido originalmente pela Ericsson, mas agora todos o utilizam.

Os padrões **de direito**, ao contrário, são padrões adotados por um órgão de padronização formal. Em geral, as autoridades de padronização internacional são divididas em duas classes: as que foram estabelecidas por tratados entre governos nacionais, e as organizações voluntárias, que foram criadas independentemente de tratados. Na área de padrões de redes de computadores, há diversas organizações de ambos os tipos, especialmente ITU, ISO, IETF e IEEE, os quais veremos nas próximas subseções.

Na prática, os relacionamentos entre padrões, empresas e órgãos de padronização são complicados. Os padrões de fato normalmente evoluem para padrões de direito, especialmente se tiverem sucesso. Isso aconteceu no caso do HTTP, que foi rapidamente adotado pelo IETF. Os órgãos de padrões normalmente ratificam os padrões mutuamente, como se estivessem dando um tapinha nas costas uns dos outros, a fim de aumentar o mercado para uma tecnologia. Atualmente, muitas alianças comerciais ocasionais que são formadas em torno de determinadas tecnologias também desempenham um papel significativo no desenvolvimento e refinamento de padrões de rede. Por exemplo, o **projeto de parceria do 3G**, ou **3GPP** (**Third Generation Partnership Project**) é uma colaboração entre associações de telecomunicações que controla os padrões de telefonia móvel 3G UMTS.

1.6.1 Quem é quem no mundo das telecomunicações

O status legal das companhias telefônicas do mundo varia consideravelmente de um país para outro. De um lado, estão os Estados Unidos, que têm mais de 2 mil empresas telefônicas separadas (em sua maioria, muito pequenas). Mais algumas foram incluídas com a divisão da AT&T em 1984 (então a maior corporação do mundo, oferecendo serviço telefônico a cerca de 80 por cento dos telefones dos Estados Unidos) e o Telecommunications Act de 1996, que reestruturou a regulamentação para promover a concorrência.

No outro extremo, estão os países em que o governo federal detém o monopólio de toda a área de comunicações, incluindo correios, telégrafos, telefone e muitas vezes rádio e televisão. A maior parte do mundo se enquadra nessa categoria. Em alguns casos, as telecomunicações são comandadas por uma empresa nacionalizada, mas em outros elas são controladas por uma estatal, em geral conhecida como administração **PTT** (**Post, Telegraph & Telephone**). No mundo inteiro, a tendência é de liberalização e competição, encerrando o monopólio do governo. A maioria dos países europeus agora tem suas PTTs (parcialmente) privatizadas, mas em outros lugares o processo ainda está ganhando impulso lentamente.

Com todos esses diferentes fornecedores de serviços, é cada vez maior a necessidade de oferecer compatibilidade em escala mundial para garantir que pessoas (e computadores) em diferentes países possam se comunicar. Na verdade, essa necessidade já existe há muito tempo. Em 1865, representantes de diversos governos europeus se reuniram para formar a predecessora da atual **ITU** (**International Telecommunication Union**). A missão da ITU era padronizar as telecomunicações internacionais, até então dominadas pelo telégrafo. Já naquela época estava claro que, se metade dos países utilizasse código Morse e a outra metade usasse algum outro código, haveria problemas de comunicação. Quando o telefone passou a ser um serviço internacional, a ITU também se encarregou de padronizar a telefonia. Em 1947, a ITU tornou-se um órgão das Nações Unidas.

A ITU tem cerca de 200 membros governamentais, incluindo quase todos os membros das Nações Unidas. Tendo em vista que os Estados Unidos não têm uma PTT, outro grupo teve de representá-los na ITU. Essa tarefa coube ao Departamento de Estado, provavelmente porque a ITU se relacionava com países estrangeiros, a especialidade desse departamento. Existem mais de setecentos membros setoriais e associados, incluindo as empresas de telefonia (por

exemplo, AT&T, Vodafone, Sprint), fabricantes de equipamentos de telecomunicações (por exemplo, Cisco, Nokia, Nortel), fornecedores de computadores (por exemplo, Microsoft, Agilent, Toshiba), fabricantes de chips (por exemplo, Intel, Motorola, TI) e outras empresas interessadas (por exemplo, Boeing, CBS, VeriSign).

A ITU tem três setores principais. Vamos nos concentrar principalmente na **ITU-T**, o setor de padronização de telecomunicações, que controla os sistemas de telefonia e de comunicação de dados. Antes de 1993, a ITU-T era conhecida como **CCITT**, acrônimo de Comité Consultatif International Télégraphique et Téléphonique, seu nome em francês. A **ITU-R**, o setor de radiocomunicações, é responsável pela coordenação do uso, por grupos de interesse concorrentes, das frequências de rádio no mundo inteiro. O outro setor é ITU-D, o setor de desenvolvimento. Ele promove o desenvolvimento de tecnologias de informação e comunicação para estreitar a 'divisão digital' entre as empresas com acesso efetivo às tecnologias de informação e países com acesso limitado.

A tarefa da ITU-T é definir recomendações técnicas para interfaces de telefonia, telégrafos e comunicação de dados. Em geral, essas recomendações se transformam em padrões internacionalmente reconhecidos, embora, tecnicamente, as recomendações da ITU-T sejam apenas sugestões que os governos podem adotar ou ignorar, como quiserem (porque os governos são como garotos de 13 anos — eles não gostam de receber ordens). Na prática, um país que deseja adotar um padrão de telefonia diferente do restante do mundo tem toda a liberdade de fazê-lo, mas ficará isolado de todos os outros. Essa opção pode ser válida na Coreia do Norte, mas seria a fonte de muitos problemas em outros lugares.

O trabalho real da ITU-T é feito em seus grupos de estudo. Atualmente existem dez grupos de estudo, geralmente com até quatrocentas pessoas, que abordam assuntos variando desde cobrança telefônica até serviços de multimídia e segurança. O SG 15, por exemplo, padroniza as tecnologias DSL popularmente usadas para conexão com a Internet. Para tornar possível a obtenção de algum resultado, os grupos de estudo se dividem em setores de trabalho que, por sua vez, se dividem em equipes de especialistas que, por sua vez, se dividem em grupos ocasionais. Uma vez burocracia, sempre burocracia.

Apesar de todas essas dificuldades, a ITU-T consegue realizar algo. Desde sua origem, ela produziu cerca de 3 mil recomendações, muitas das quais são amplamente utilizadas na prática. Por exemplo, a Recomendação H.264 (também um padrão ISO conhecido como MPEG-4 AVC) é bastante usada para compactação de vídeo, e os certificados de chave pública X.509 são usados para navegação segura na Web e para assinaturas digitais no correio eletrônico.

Quando a transição iniciada na década de 1980 for concluída e as telecomunicações deixarem de ser uma questão interna de cada país para ganhar o status de questão global, os padrões ganharão cada vez mais importância, e um número cada vez maior de organizações desejará participar do processo de definição de padrões. Para obter mais informações sobre a ITU, consulte Irmer (1994).

1.6.2 Quem é quem no mundo dos padrões internacionais

Os padrões internacionais são produzidos e publicados pela **ISO** (**International Standards Organization**[†]), uma organização voluntária independente, fundada em 1946. Seus membros são as organizações nacionais de padrões dos 157 países membros. Dentre eles estão as seguintes organizações: ANSI (Estados Unidos), BSI (Grã-Bretanha), AFNOR (França), DIN (Alemanha) e 153 outros.

A ISO publica padrões sobre uma vasta gama de assuntos, desde porcas e parafusos (literalmente) ao revestimento usado nos postes telefônicos [sem mencionar sementes de cacau (ISO 2451), redes de pesca (ISO 1530), roupas íntimas femininas (ISO 4416) e vários outros assuntos que ninguém imaginaria que estivessem sujeitos à padronização]. Em questões de padrões de telecomunicação, a ISO e a ITU-T normalmente cooperam (a ISO é um membro da ITU-T) para evitar a ironia de dois padrões internacionais oficiais e mutuamente incompatíveis.

A ISO já publicou mais de 17 mil padrões, incluindo os padrões OSI. Ela tem mais de duzentos comitês técnicos, ou TCs (Technical Committees), numerados por ordem de criação, cada um lidando com um assunto específico. O TC1 lida com porcas e parafusos (padronizando as medidas da rosca). O JTC1 trata da tecnologia de informação, incluindo redes, computadores e software. Ele é o primeiro (e até aqui único) comitê técnico conjunto, criado em 1987 mesclando o TC97 com as atividades no IEC, outro órgão de padronização. Cada TC tem subcomitês (SCs) que, por sua vez, se dividem em grupos de trabalho (*working groups* — WGs).

O trabalho real da ISO é feito em grande parte nos grupos de trabalho, em torno dos quais se reúnem 100 mil voluntários de todo o mundo. Muitos desses 'voluntários' foram escalados para trabalhar em questões da ISO pelos seus empregadores, cujos produtos estão sendo padronizados. Outros são funcionários públicos ansiosos por descobrir um meio de transformar o que é feito em seus países de origem em padrão internacional. Especialistas acadêmicos também têm participação ativa em muitos grupos de trabalho.

O procedimento usado pela ISO para a adoção de padrões foi criado de modo a obter o maior consenso possível. O processo começa quando uma das organizações de padrões nacionais sente a necessidade de um padrão internacional em alguma área. Em seguida, é formado um grupo de trabalho com a finalidade de produzir um **ras-**

[†] Para os puristas, o verdadeiro nome é International Organization for Standardization.

cunho de comitê ou **CD** (**Committee Draft**). Depois, o CD é distribuído a todas as entidades associadas, que têm seis meses para analisá-lo. Se ele for aprovado por uma ampla maioria, um documento revisado, chamado **rascunho de norma internacional** ou **DIS** (**Draft International Standard**), será produzido e distribuído para receber comentários e ser votado. Com base nos resultados dessa rodada, o texto final do **padrão internacional** ou **IS** (**International Standard**) é preparado, aprovado e publicado. Nas áreas de grande controvérsia, o CD ou o DIS passam por diversas revisões até obter o número de votos necessário, em um processo que pode durar anos.

O **NIST** (**National Institute of Standards and Technology**) é um órgão do Departamento de Comércio dos Estados Unidos. Ele, que já foi chamado de National Bureau of Standards, emite padrões que controlam as compras feitas pelo governo dos Estados Unidos, exceto as do Departamento de Defesa, que tem seus próprios padrões.

Outro participante essencial no mundo dos padrões é o **IEEE** (**Institute of Electrical and Electronics Engineers**), a maior organização profissional do mundo. Além de publicar uma série de jornais e promover diversas conferências a cada ano, o IEEE tem um grupo de padronização que desenvolve padrões nas áreas de engenharia elétrica e informática. O comitê 802 do IEEE padronizou vários tipos de LANs. Estudaremos alguns de seus resultados mais adiante neste livro. O trabalho é feito por um conjunto de grupos de trabalho, os quais estão listados na Tabela 1.4. A taxa de sucesso dos diversos grupos de trabalho do 802 tem sido baixa; ter um número 802.x não é garantia de sucesso. Porém, o impacto das histórias de sucesso (em especial do 802.3 e do 802.11) no setor e no mundo tem sido enorme.

Número	Assunto
802.1	Avaliação e arquitetura de LANs
802.2 ↓	Controle de link lógico
802.3 *	Ethernet
802.4 ↓	Token bus (barramento de tokens; foi usado por algum tempo em unidades industriais)
802.5	Token ring (anel de tokens; a entrada da IBM no mundo das LANs)
802.6 ↓	Fila dual barramento dual (primeira rede metropolitana)
802.7 ↓	Grupo técnico consultivo sobre tecnologias de banda larga
802.8 †	Grupo técnico consultivo sobre tecnologias de fibra óptica
802.9 ↓	LANs isócronas (para aplicações em tempo real)
802.10 ↓	LANs virtuais e segurança
802.11 *	LANs sem fios (WiFi)
802.12 ↓	Prioridade de demanda (AnyLAN da Hewlett-Packard)
802.13	Número relacionado à má sorte. Ninguém o quis
802.14 ↓	Modems a cabo (extinto: um consórcio industrial conseguiu chegar primeiro)
802.15 *	Redes pessoais (Bluetooth, Zigbee)
802.16 *	Banda larga sem fio (WiMAX)
802.17	Anel de pacote resiliente
802.18	Grupo técnico consultivo sobre questões de regulamentação de rádio
802.19	Grupo técnico consultivo sobre coexistência de todos esses padrões
802.20	Banda larga móvel sem fio (semelhante ao 802.16e)
802.21	Transferência independente do meio (para tecnologias de roaming)
802.22	Rede regional sem fios

Tabela 1.4 | Os grupos de trabalho 802. Os grupos importantes estão marcados com *. Aqueles marcados com ↓ estão hibernando. O grupo marcado com † desistiu e foi dissolvido.

1.6.3 Quem é quem no mundo dos padrões da Internet

A Internet mundial tem seus próprios mecanismos de padronização, que são bastante diferentes dos adotados pela ITU-T e pela ISO. *Grosso modo*, pode-se dizer que as pessoas que participam das reuniões de padronização da ITU ou da ISO se apresentam de paletó e gravata, ao passo que as pessoas que participam das reuniões de padronização na Internet usam jeans (exceto quando os encontros são em locais quentes, quando utilizam bermudas e camisetas).

As reuniões da ITU-T e da ISO são frequentadas por pessoas ligadas à iniciativa privada e ao governo, cuja especialidade é a padronização. Para essas pessoas, a padronização é algo sagrado e a ela dedicam suas vidas. Por outro lado, as pessoas ligadas à Internet têm uma natureza anárquica. No entanto, com centenas de milhões de pessoas fazendo tudo por sua conta, pode haver uma pequena comunicação. Por essa razão, os padrões — apesar dos pesares — acabam se fazendo necessários. Neste contexto, David Clark fez o seguinte comentário sobre padronização na Internet, hoje famoso: 'consenso rígido e código funcional'.

Quando a ARPANET foi configurada, o Departamento de Defesa dos Estados Unidos criou um comitê informal para supervisioná-la. Em 1983, o comitê passou a ser chamado **IAB** (**Internet Activities Board**) e teve seus objetivos ampliados, ou seja, foi possível manter os pesquisadores envolvidos com a ARPANET e a Internet mais ou menos voltados para uma mesma direção, uma tarefa bastante árdua, diga-se de passagem. Mais tarde, o significado do acrônimo 'IAB' mudou para **Internet Architecture Board**.

Cada um dos cerca de dez membros do IAB coordenou uma força-tarefa sobre alguma questão importante. O IAB promovia diversas reuniões anuais para discutir os resultados e prestar contas ao Departamento de Defesa e à NSF, que na época estavam financiando a maior parte de suas atividades. Quando havia necessidade de um padrão (por exemplo, um novo algoritmo de roteamento), os membros do IAB o elaboravam e, em seguida, anunciavam a mudança aos estudantes universitários, de modo que os envolvidos na produção do software pudessem implementá-lo. A comunicação era feita por uma série de relatórios técnicos, chamados **RFCs** (**Request For Comments**). As RFCs são armazenadas on-line, e todas as pessoas interessadas podem ter acesso a elas em www.ietf.org/rfc. Elas são numeradas em ordem cronológica de criação, e já estão na casa dos 5 mil. Vamos nos referir a muitas RFCs neste livro.

Por volta de 1989, a Internet havia crescido tanto que teve de abdicar de seu estilo altamente informal. Muitos fabricantes estavam oferecendo produtos TCP/IP e não queriam mudá-los só porque uma dezena de pesquisadores acreditava ter uma ideia melhor. No verão de 1989, o IAB se reorganizou mais uma vez. Os pesquisadores se reuniram em torno da **IRTF** (**Internet Research Task Force**), que se transformou em uma subsidiária do IAB, junto com a **IETF** (**Internet Engineering Task Force**). O IAB foi novamente ocupado por pessoas que representavam uma faixa mais ampla de organizações que a simples comunidade de pesquisa. Inicialmente, os membros do grupo teriam um mandato indireto de dois anos, sendo os novos membros indicados pelos antigos. Mais tarde foi criada a **Internet Society**, integrada por pessoas interessadas na Internet. De certa forma, a Internet Society pode ser comparada à ACM ou ao IEEE. Ela é administrada por conselheiros eleitos que, por sua vez, indicam os membros do IAB.

A ideia dessa divisão era fazer com que a IRTF se concentrasse em pesquisas a longo prazo, enquanto a IETF lidaria com questões de engenharia a curto prazo. A IETF foi dividida em grupos de trabalho, e cada um deveria resolver um problema específico. Os coordenadores desses grupos de trabalho inicialmente formariam uma espécie de comitê geral para orientar o esforço de engenharia. Dentre os assuntos estudados pelos grupos de trabalho estão novas aplicações, informações para o usuário, integração do OSI, roteamento e endereçamento, segurança, gerenciamento de redes e padrões. Por fim, formaram-se tantos grupos de trabalho (mais de 70) que foi necessário agrupá-los em áreas, cujos coordenadores passaram a integrar o comitê geral.

Além disso, foi adotado um processo de padronização mais formal, semelhante aos da ISO. Para se tornar um **Proposed Standard** (**padrão proposto**), a ideia básica deve ser explicada em uma RFC e despertar na comunidade um interesse suficiente para merecer algum tipo de consideração. Para chegar ao estágio de **Draft Standard** (**padrão de rascunho**), o padrão proposto precisa ser rigorosamente testado por, no mínimo, dois locais independentes durante quatro meses. Se o IAB for convencido de que a ideia é viável e de que o software funciona, ele poderá atribuir à RFC em questão o status de **Internet Standard** (**padrão da Internet**). Alguns padrões da Internet foram adotados pelo Departamento de Defesa dos Estados Unidos (MIL-STD), tornando-se obrigatórios para seus fornecedores.

Para os padrões da Web, o **World Wide Web Consortium** (**W3C**) desenvolve protocolos e diretrizes para facilitar o crescimento da Web a longo prazo. Esse é um consórcio industrial liderado por Tim Berners-Lee e estabelecido em 1994, quando a Web realmente começou a ganhar força. O W3C agora tem mais de trezentos membros do mundo inteiro e já produziu mais de cem Recomendações do W3C, como são chamados seus padrões, abordando assuntos como HTML e privacidade na Web.

1.7 Unidades de medida

Para evitar qualquer confusão, vale a pena declarar explicitamente que, neste livro, como na ciência da computação em geral, as unidades de medida são usadas no lugar das unidades tradicionais inglesas. Os principais prefixos de medida estão listados na Tabela 1.5. Em geral, os prefixos são abreviados por sua letra inicial, com as unidades maiores que 1 em maiúsculas (KB, MB etc.). Uma exceção (por razões históricas) é a unidade kbps para indicar kilobits/s. Desse modo, uma linha de comunicação de 1 Mbps transmite 10^6 bits/s e um clock de 100 psegundos (ou 100 ps) pulsa a cada 10^{-10} segundos. Tendo em vista que os prefixos mili e micro começam ambos pela letra 'm', foi preciso fazer uma escolha. Normalmente, 'm' representa mili e 'μ' (a letra grega mi) representa micro.

Também vale a pena assinalar que, para medir tamanhos de memória, disco, arquivos e bancos de dados, uma prática comum na indústria, as unidades têm significados um pouco diferentes. Nesses casos, kilo significa 2^{10} (1.024), em vez de 10^3 (1.000), porque as memórias sempre são medidas em potências de dois. Desse modo, uma memória de 1 KB contém 1.024 bytes, e não 1.000 bytes. Observe também que a letra 'B' maiúscula, nesse uso, significa 'bytes' (unidades de oito bits), enquanto uma letra 'b' minúscula significa 'bits'. De modo semelhante, uma memória de 1 MB contém 2^{20} (1.048.576) bytes, uma memória de 1 GB contém 2^{30} (1.073.741.824) bytes e um banco de dados de 1 TB contém 2^{40} (1.099.511.627.776) bytes. No entanto, uma linha de comunicação de 1 kbps transmite 1.000 bits por segundo, e uma LAN de 10 Mbps funciona a 10.000.000 bits/s, porque essas velocidades não são potências de dois. Infelizmente, muitas pessoas tendem a misturar esses dois sistemas, especialmente para tamanhos de disco. Para evitar ambiguidade, neste livro usaremos os símbolos KB, MB, GB e TB para 2^{10}, 2^{20}, 2^{30} e 2^{40} bytes, respectivamente, e os símbolos kbps, Mbps, Gbps e Tbps para 10^3, 10^6, 10^9 e 10^{12} bits/s, respectivamente.

1.8 Visão geral dos outros capítulos do livro

Este livro descreve os princípios e a prática em redes de computadores. A maioria dos capítulos começa com uma discussão dos princípios relevantes, seguida por uma série de exemplos ilustrativos. Em geral, esses exemplos são extraídos da Internet e das redes sem fios, como a rede de telefonia móvel, uma vez que tais redes são importantes e muito diferentes. Serão apresentados outros exemplos quando for relevante.

A estrutura deste livro segue o modelo híbrido da Figura 1.20. A partir do Capítulo 2, vamos começar a percorrer nosso caminho pela hierarquia de protocolos, começando pela parte inferior. Apresentaremos uma rápida análise do processo de comunicação de dados, com sistemas de transmissão cabeada e sem fio. Esse material está voltado para a camada física, apesar de examinarmos apenas sua arquitetura e deixarmos de lado os aspectos de hardware. Diversos exemplos da camada física também são discutidos, como as redes de telefonia pública comutada, de telefones celulares e a rede de televisão a cabo.

Os capítulos 3 e 4 discutem a camada de enlace de dados em duas partes. O Capítulo 3 examina o problema de como enviar pacotes por um enlace, incluindo detecção e correção de erros. Examinamos o DSL (usado para acesso à Internet de banda larga por linhas telefônicas) como um exemplo do mundo real de um protocolo de enlace de dados.

O Capítulo 4 é dedicado à subcamada de acesso ao meio, que faz parte da camada de enlace de dados, que lida com a questão de como compartilhar um canal entre vários computadores. Os exemplos que examinamos incluem redes sem fios, como 802.11 e RFID, e LANs com fio, como a Ethernet clássica. Também discutimos os switches da camada de enlace que conectam as LANs, como a Ethernet comutada.

Exp.	Explícita	Prefixo	Exp.	Explícita	Prefixo
10^{-3}	0,001	mili	10^{3}	1.000	Kilo
10^{-6}	0,000001	micro	10^{6}	1.000.000	Mega
10^{-9}	0,000000001	nano	10^{9}	1.000.000.000	Giga
10^{-12}	0,000000000001	pico	10^{12}	1.000.000.000.000	Tera
10^{-15}	0,000000000000001	femto	10^{15}	1.000.000.000.000.000	Peta
10^{-18}	0,000000000000000001	atto	10^{18}	1.000.000.000.000.000.000	Exa
10^{-21}	0,000000000000000000001	zepto	10^{21}	1.000.000.000.000.000.000.000	Zetta
10^{-24}	0,000000000000000000000001	yocto	10^{24}	1.000.000.000.000.000.000.000.000	Yotta

Tabela 1.5 | Os principais prefixos de medida.

O Capítulo 5 trata da camada de rede, em especial o roteamento. Serão abordados muitos algoritmos de roteamento, tanto estático quanto dinâmico. Porém, mesmo com bons algoritmos de roteamento, se for oferecido mais tráfego do que a rede pode manipular, alguns pacotes sofrerão atrasos ou serão descartados. Discutimos essa questão desde como impedir o congestionamento até como garantir uma certa qualidade de serviço. A conexão de redes heterogêneas para formar redes interligadas leva a numerosos problemas que também serão analisados. A camada de rede na Internet recebe uma extensa abordagem.

O Capítulo 6 é dedicado à camada de transporte. Grande parte da ênfase é sobre os protocolos orientados a conexões e confiança, uma vez que muitas aplicações necessitam deles. Estudaremos em detalhes os protocolos de transporte da Internet, UDP e TCP, bem como seus problemas de desempenho.

O Capítulo 7 é dedicado à camada de aplicação, seus protocolos e suas aplicações. O primeiro tópico é o DNS, que é o catálogo telefônico da Internet. Em seguida, vem o correio eletrônico, incluindo uma discussão de seus protocolos. Depois, passamos para a Web, com descrições detalhadas de conteúdo estático e dinâmico, e o que acontece nos lados cliente e servidor. Depois disso, examinamos multimídia em rede, incluindo streaming de áudio e vídeo. Por fim, discutimos as redes de entrega de conteúdo, incluindo a tecnologia peer-to-peer.

O Capítulo 8 dedica-se à segurança das redes. Esse tópico tem aspectos que se relacionam a todas as camadas; assim, é mais fácil estudá-los depois que todas as camadas tiverem sido completamente examinadas. O capítulo começa com uma introdução à criptografia. Mais adiante, o capítulo mostra como a criptografia pode ser usada para garantir a segurança da comunicação, do correio eletrônico e da Web. O capítulo termina com uma discussão de algumas áreas em que a segurança atinge a privacidade, a liberdade de expressão, a censura e outras questões sociais.

O Capítulo 9 contém uma lista comentada de leituras sugeridas, organizadas por capítulo. Seu objetivo é ajudar os leitores que desejam ter mais conhecimentos sobre redes. O capítulo também apresenta uma bibliografia em ordem alfabética com todas as referências citadas neste livro.

1.9 Resumo

As redes de computadores têm inúmeros usos, tanto por empresas quanto por indivíduos, tanto em casa quanto em trânsito. Para as empresas, as redes de computadores pessoais que utilizam servidores compartilhados frequentemente oferecem acesso a informações corporativas, normalmente usando o modelo cliente-servidor com os desktops de funcionários atuando como clientes que acessam servidores poderosos na sala de máquinas. Para as pessoas, as redes oferecem acesso a uma série de informações e fontes de entretenimento, bem como um modo de comprar e vender produtos e serviços. Em geral, as pessoas têm acesso à Internet com a utilização de um telefone ou provedores a cabo em casa, embora um número cada vez maior de pessoas tenha uma conexão sem fio para laptops e telefones. Os avanços na tecnologia estão permitindo novos tipos de aplicações móveis e redes com computadores embutidos em aparelhos e outros dispositivos do usuário. Os mesmos avanços levantam questões sociais, como preocupações acerca de privacidade.

Grosso modo, as redes podem ser divididas em LANs, MANs, WANs e internets. As LANs abrangem um prédio e operam em altas velocidades. As MANs em geral abrangem uma cidade, por exemplo, o sistema de televisão a cabo, que hoje é utilizado por muitas pessoas para acessar a Internet. As WANs abrangem um país ou um continente. Algumas das tecnologias usadas para montar essas redes são ponto a ponto (por exemplo, um cabo), enquanto outras são por broadcast (por exemplo, as redes sem fio). As redes podem ser interconectadas com roteadores para formar internets, das quais a Internet é o exemplo maior e mais conhecido. As redes sem fios, por exemplo, as LANs 802.11 e a telefonia móvel 3G, também estão se tornando extremamente populares.

O software de rede consiste em protocolos ou regras pelas quais os processos se comunicam. A maioria das redes aceita as hierarquias de protocolos, com cada camada fornecendo serviços às camadas situadas acima dela e isolando-as dos detalhes dos protocolos usados nas camadas inferiores. Em geral, as pilhas de protocolos se baseiam no modelo OSI ou no TCP/IP. Esses dois modelos têm camadas de enlace, rede, transporte e aplicação, mas apresentam diferenças nas outras camadas. As questões de projeto incluem confiabilidade, alocação de recursos, crescimento, segurança e outros. Grande parte deste livro lida com protocolos e seu projeto.

As redes fornecem vários serviços a seus usuários. Esses serviços podem variar da entrega de pacotes por melhores esforços por serviços não orientados a conexões até a entrega garantida por serviços orientados a conexões. Em algumas redes, o serviço não orientado a conexões é fornecido em uma camada e o serviço orientado a conexões é oferecido na camada acima dela.

Entre as redes mais conhecidas estão a Internet, a rede de telefonia móvel 3G e as LANs 802.11. A Internet evoluiu a partir da ARPANET, à qual foram acrescentadas outras redes para formar uma rede interligada. A Internet atual é, na realidade, um conjunto com muitos milhares de redes que usam a pilha de protocolos TCP/IP. A rede de telefonia móvel 3G oferece acesso sem fio e móvel à Internet, em velocidades de múltiplos Mbps e, naturalmente, também realiza chamadas de voz. As LANs sem fios baseadas no padrão IEEE 802.11 são implantadas em muitas casas e cafés, e podem oferecer co-

nectividade em velocidades superiores aos 100 Mbps. Novos tipos de redes também estão surgindo, como as redes de sensores embutidas e as redes baseadas na tecnologia RFID.

Fazer vários computadores se comunicarem exige uma extensa padronização, tanto de hardware quanto de software. Organizações como ITU-T, ISO, IEEE e IAB administram partes diferentes do processo de padronização.

Problemas

1. Imagine que você tenha treinado Bernie, seu cachorro São Bernardo, para carregar uma caixa de três fitas de 8 mm, em vez de um cantil de conhaque. (Quando seu disco ficar cheio, considere isso como uma emergência.) Cada uma dessas fitas contém 7 gigabytes. O cachorro pode viajar a seu lado, onde quer que você esteja, a 18 km/h. Para qual intervalo de distâncias Bernie terá uma taxa de dados mais alta que uma linha de transmissão cuja taxa de dados (excluindo o overhead) é de 150 Mbps? Como sua resposta mudaria se (i) A velocidade de Bernie dobrasse; (ii) a capacidade de cada fita fosse dobrada; (iii) a taxa de dados da linha de transmissão fosse dobrada.
2. Uma alternativa para uma LAN é simplesmente instalar um grande sistema de tempo compartilhado (timesharing) com terminais para todos os usuários. Apresente duas vantagens de um sistema cliente-servidor que utilize uma LAN.
3. O desempenho de um sistema cliente-servidor é influenciado por dois fatores de rede: a largura de banda da rede (quantos bits/s ela pode transportar) e a latência (quantos segundos o primeiro bit leva para ir do cliente até o servidor). Dê um exemplo de uma rede que exibe alta largura de banda e alta latência. Depois, dê um exemplo de rede com baixa largura de banda e baixa latência.
4. Além da largura de banda e da latência, que outro parâmetro é necessário para permitir uma boa caracterização da qualidade de serviço oferecida por uma rede empregada para (i) tráfego de voz digitalizada? (ii) tráfego de vídeo? (iii) tráfego de transação financeira?
5. Um fator que influencia no atraso de um sistema de comutação de pacotes store-and-forward é qual o tempo necessário para armazenar e encaminhar um pacote por um switch. Se o tempo de comutação é 10 μs, é provável que esse seja um fator importante na resposta de um sistema cliente-servidor quando o cliente está em Nova York e o servidor está na Califórnia? Suponha que a velocidade de propagação em cobre e fibra seja igual a 2/3 da velocidade da luz no vácuo.
6. Um sistema cliente-servidor usa uma rede de satélite, com o satélite a uma altura de 40.000 km. Qual é o maior atraso em resposta a uma solicitação?
7. No futuro, quando todo mundo tiver um terminal doméstico conectado a uma rede de computadores, será possível realizar plebiscitos instantâneos sobre questões importantes. É provável que a política atual seja eliminada, permitindo que as pessoas expressem seus desejos de maneira mais direta. Os aspectos positivos dessa democracia direta são óbvios; analise alguns dos aspectos negativos.
8. Um conjunto de cinco roteadores deve ser conectado a uma sub-rede ponto a ponto. Entre cada par de roteadores, os projetistas podem colocar uma linha de alta velocidade, uma linha de média velocidade, uma linha de baixa velocidade ou nenhuma linha. Se forem necessários 100 ms do tempo do computador para gerar e inspecionar cada topologia, quanto tempo será necessário para inspecionar todas elas?
9. Uma desvantagem de uma sub-rede de broadcast é a capacidade desperdiçada quando há vários hosts tentando acessar o canal ao mesmo tempo. Como um exemplo simples, suponha que o tempo esteja dividido em slots discretos, com cada um dos n hosts tentando usar o canal com probabilidade p durante cada slot. Que fração dos slots é desperdiçada em consequência das colisões?
10. Quais são as duas razões para a utilização de protocolos dispostos em camadas? Qual é uma possível desvantagem do uso de protocolos dispostos em camadas?
11. O presidente da Specialty Paint Corp. resolve trabalhar com uma cervejaria local com a finalidade de produzir uma lata de cerveja invisível (como uma medida para evitar acúmulo de lixo). O presidente pede que o departamento jurídico analise a questão e este, por sua vez, entra em contato com a empresa de engenharia. Como resultado, o engenheiro-chefe entra em contato com o funcionário de cargo equivalente na cervejaria para discutir os aspectos técnicos do projeto. Em seguida, os engenheiros enviam um relatório a seus respectivos departamentos jurídicos, que então discutem por telefone os aspectos legais. Por fim, os presidentes das duas empresas discutem as questões financeiras do negócio. Que princípio de um protocolo multicamadas (com base no modelo OSI) esse mecanismo de comunicação infringe?
12. Duas redes podem oferecer um serviço orientado a conexões bastante confiável. Uma delas oferece um fluxo de bytes confiável e a outra, um fluxo de mensagens confiável. Elas são idênticas? Se forem, por que se faz essa distinção? Se não, dê um exemplo de como elas diferem.
13. O que significa 'negociação' em uma discussão sobre protocolos de rede? Dê um exemplo.
14. Na Figura 1.16, é mostrado um serviço. Há outros serviços implícitos nessa figura? Em caso afirmativo, onde? Caso contrário, por que não?
15. Em algumas redes, a camada de enlace de dados trata os erros de transmissão solicitando a retransmissão de quadros danificados. Se a probabilidade de um quadro estar danificado é p, qual é o número médio de transmissões necessárias para enviar um quadro? Suponha que as confirmações nunca sejam perdidas.

16. Um sistema tem uma hierarquia de protocolos com n camadas. As aplicações geram mensagens com M bytes de comprimento. Em cada uma das camadas é acrescentado um cabeçalho com h bytes. Que fração da largura de banda da rede é preenchida pelos cabeçalhos?
17. Qual é a principal diferença entre o TCP e o UDP?
18. A sub-rede da Figura 1.22(b) foi projetada para resistir a uma guerra nuclear. Quantas bombas seriam necessárias para particionar os nós em dois conjuntos desconectados? Suponha que qualquer bomba destrua um nó e todos os links conectados a ele.
19. A cada 18 meses, a Internet praticamente dobra de tamanho. Embora ninguém possa dizer com certeza, estima-se que havia 600 milhões de hosts em 2009. Utilize esses dados para calcular o número previsto de hosts da Internet em 2018. Você acredita nisso? Explique por que ou por que não.
20. Quando um arquivo é transferido entre dois computadores, duas estratégias de confirmação são possíveis. Na primeira, o arquivo é dividido em pacotes, os quais são confirmados individualmente pelo receptor, mas a transferência do arquivo como um todo não é confirmada. Na segunda, os pacotes não são confirmados individualmente, mas, ao chegar a seu destino, o arquivo inteiro é confirmado. Analise essas duas abordagens.
21. As operadoras da rede de telefonia móvel precisam saber onde os telefones móveis de seus assinantes (logo, seus usuários) estão localizados. Explique por que isso é ruim para os usuários. Agora, dê motivos pelos quais isso é bom para os usuários.
22. Qual era o comprimento de um bit, em metros, no padrão 802.3 original? Utilize uma velocidade de transmissão de 10 Mbps e suponha que a velocidade de propagação no cabo coaxial seja igual a 2/3 da velocidade da luz no vácuo.
23. Uma imagem tem 1.600 × 1.200 pixels com 3 bytes/pixel. Suponha que a imagem seja descompactada. Quanto tempo é necessário para transmiti-la por um canal de modem de 56 kbps? E por um modem a cabo de 1 Mbps? E por uma rede Ethernet de 10 Mbps? E pela rede Ethernet de 100 Mbps? E pela gigabit Ethernet?
24. A Ethernet e as redes sem fios apresentam algumas semelhanças e algumas diferenças. Uma propriedade da Ethernet é de que apenas um quadro pode ser transmitido de cada vez em uma rede Ethernet. A rede 802.11 compartilha essa propriedade com a Ethernet? Analise sua resposta.
25. Liste duas vantagens e duas desvantagens da existência de padrões internacionais para protocolos de redes.
26. Quando um sistema tem uma parte permanente e uma parte removível (como uma unidade de CD-ROM e o CD-ROM), é importante que o sistema seja padronizado, de modo que empresas diferentes possam produzir as partes permanentes e as removíveis, para que sejam compatíveis entre si. Cite três exemplos fora da indústria de informática em que esses padrões internacionais estão presentes. Agora, cite três áreas fora da indústria de informática em que eles não estão presentes.
27. Suponha que os algoritmos usados para implementar as operações na camada k sejam mudados. Como isso afeta as operações nas camadas $k - 1$ e $k + 1$?
28. Suponha que haja uma mudança no serviço (conjunto de operações) fornecido pela camada k. Como isso afeta os serviços nas camadas $k - 1$ e $k + 1$?
29. Faça uma lista de motivos pelos quais o tempo de resposta de um cliente pode ser maior do que o atraso no melhor caso.
30. Quais são as desvantagens do uso de células pequenas, de tamanho fixo, em ATM?
31. Faça uma lista de atividades que você pratica todo dia em que são utilizadas redes de computadores. Como sua vida seria alterada se de repente essas redes fossem desativadas?
32. Descubra quais são as redes usadas em sua escola ou local de trabalho. Descreva os tipos de redes, as topologias e os métodos de comutação utilizados.
33. O programa *ping* lhe permite enviar um pacote de teste a um dado local e verificar quanto tempo ele demora para ir e voltar. Tente usar *ping* para ver quanto tempo ele demora para trafegar do local em que você esta até vários locais conhecidos. A partir desses dados, represente o tempo em trânsito de mão única pela Internet como uma função de distância. É melhor usar universidades, uma vez que a localização de seus servidores é conhecida com grande precisão. Por exemplo, berkeley.edu está em Berkeley, Califórnia; mit.edu está em Cambridge, Massachusetts; vu.nl está em Amsterdã, Holanda; www.usyd.edu.au está em Sydney, Austrália; e www.uct.ac.za está em Cidade do Cabo, África do Sul.
34. Vá ao website da IETF, www.ietf.org, ver o que eles estão fazendo. Escolha um projeto de que você goste e escreva um relatório de meia página sobre o problema e a solução proposta.
35. A Internet é composta por um grande número de redes. Sua organização determina a topologia da Internet. Uma quantidade considerável de informações sobre a topologia da Internet está disponível on-line. Use um mecanismo de busca para descobrir mais sobre a topologia da Internet e escreva um breve relatório resumindo suas descobertas.
36. Pesquise na Internet para descobrir alguns dos pontos de emparelhamento (peering points) importantes, usados atualmente para o roteamento de pacotes na Internet.
37. Escreva um programa que implemente o fluxo de mensagens da camada superior até a camada inferior do modelo de protocolo de sete camadas. Seu programa deverá incluir uma função de protocolo separada para cada camada. Os cabeçalhos de protocolo são sequências de até 64 caracteres. Cada função do protocolo tem dois parâmetros: uma mensagem passada do protocolo da camada mais alta (um char buffer) e o tamanho da mensagem. Essa função conecta seu cabeçalho na frente da mensagem, imprime a nova mensagem na saída-padrão e depois chama a função do protocolo da camada inferior. A entrada do programa é uma mensagem de aplicação (uma sequência de 80 caracteres ou menos).

A camada física

Capítulo 2

Neste capítulo, analisaremos a camada mais baixa da hierarquia em nosso modelo de protocolo. Ela define as interfaces elétrica, de sincronização e outras, pelas quais os bits são enviados como sinais pelos canais. A camada física é o alicerce sobre o qual a rede é construída. Como as propriedades dos diferentes tipos de canais físicos determinam o desempenho (por exemplo, troughput, latência e taxa de erros), este é um bom lugar para começar nossa jornada até a 'terra das redes'.

Inicialmente, faremos uma análise teórica da transmissão de dados, apenas para descobrir que a Mãe Natureza impõe limites sobre o que pode ser enviado por um canal. Em seguida, discutiremos três meios de transmissão: guiado (fio de cobre e fibra óptica), sem fio (rádio terrestre) e satélite. Cada uma dessas tecnologias tem diferentes propriedades que afetam o projeto e o desempenho das redes que as utilizam. Esse material fornecerá informações fundamentais sobre as principais tecnologias de transmissão usadas em redes modernas.

Em seguida, abordaremos a modulação digital, que trata de como os sinais analógicos são convertidos em bits digitais e a sinais novamente. A seguir, examinaremos os esquemas de multiplexação, explorando como várias conversas podem ser feitas no mesmo meio de transmissão ao mesmo tempo, sem interferir umas com as outras.

Por fim, veremos três exemplos de sistemas de comunicação usados na prática nas redes de computadores a longas distâncias: o sistema de telefonia (fixa), o sistema de telefonia móvel (ou celular) e o sistema de televisão a cabo. Como os três são muito importantes na prática, dedicaremos uma boa quantidade de espaço a cada um.

2.1 A base teórica da comunicação de dados

As informações podem ser transmitidas por fios, fazendo-se variar alguma propriedade física, como tensão ou corrente. Representando o valor dessa tensão ou corrente como uma função de tempo com um valor único, $f(t)$, podemos criar um modelo para o comportamento do sinal e analisá-lo matematicamente. Essa análise será o assunto das próximas seções.

2.1.1 Análise de Fourier

No início do século XIX, o matemático francês Jean-Baptiste Fourier provou que qualquer função periódica razoavelmente estável, $g(t)$, com período T, pode ser construída como a soma de um número (possivelmente infinito) de senos e cossenos:

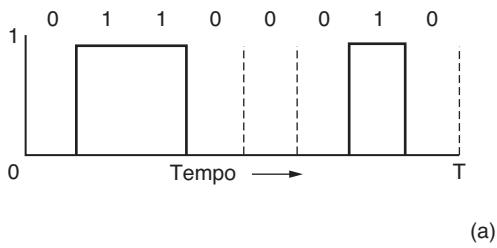
$$g(t) = \frac{1}{2}c + \sum_{n=1}^{\infty} a_n sen(\pi nft) + \sum_{n=1}^{\infty} b_n \cos(2\pi nft) \qquad (2.1)$$

onde $f = 1/T$ é a frequência fundamental, a_n e b_n são as amplitudes do seno e do cosseno dos n-ésimos **harmônicos** (termos) e c é uma constante. Essa decomposição é chamada **série de Fourier**. A partir da série de Fourier, a função pode ser reconstruída, ou seja, se o período T for conhecido e as amplitudes forem dadas, a função original no tempo poderá ser encontrada efetuando-se as somas da Equação 2.1.

Um sinal de dados com uma duração finita (como acontece com todos eles) pode ser tratado apenas com base na premissa de que ele repete o mesmo padrão (ou seja, o intervalo de T a $2T$ é igual ao de 0 a T etc.).

As a_n amplitudes podem ser calculadas para qualquer $g(t)$ dada, multiplicando-se ambos os lados da Equação 2.1 por $sen(2\pi kft)$ e, em seguida, integrando-se de 0 a T. Como

$$\int_0^T sen(2\pi kft) sen(2\pi nft)\, dt = \begin{cases} 0 \text{ para } k \neq n \\ T/2 \text{ para } k = n \end{cases}$$

apenas um termo do somatório permanece: a_n. O somatório b_n desaparece completamente. Da mesma forma, multiplicando a Equação 2.1 por $\cos(2\pi kft)$ e integrando entre 0 e T, podemos derivar b_n. Integrando ambos os lados da equação tal como ela se encontra, podemos achar c. Os resultados dessas operações são:

$$a_n = \frac{2}{T}\int_0^T g(t) sen(2\pi nft)\, dt$$

$$b_n = \frac{2}{T}\int_0^T g(t)\cos(2\pi nft)\, dt$$

$$c = \frac{2}{T}\int_0^T g(t)\, dt$$

2.1.2 Sinais limitados pela largura de banda

A importância de tudo isso para a comunicação de dados é que os canais reais afetam diferentes sinais de frequência de formas diferentes. Vamos analisar um exemplo específico: a transmissão do caractere ASCII 'b' codificado como um byte de 8 bits. O padrão de bits que deve ser transmitido é 01100010. A parte esquerda da Figura 2.1(a) mostra a saída de tensão do computador transmissor. A análise de Fourier desse sinal produz os seguintes coeficientes:

$$a_n = \frac{1}{\pi n}[\cos(\pi n/4) - \cos(3\pi/4) + \cos(6\pi n/4) - \cos(7\pi n/4)]$$

$$b_n = \frac{1}{\pi n}[sen(3\pi/4) - sen(\pi n/4) + sen(7\pi n/4) - sen(6\pi n/4)]$$

$$c = 3/4$$

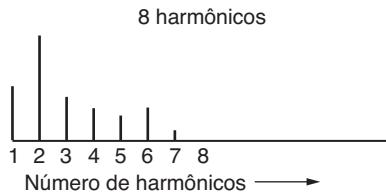
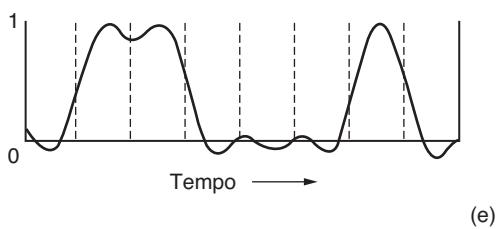
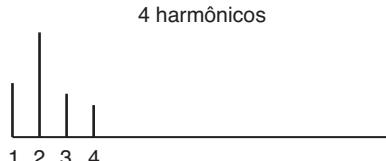
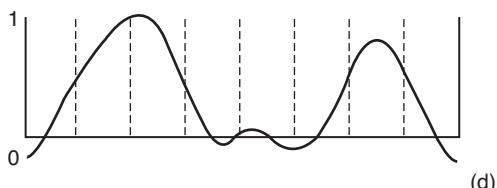
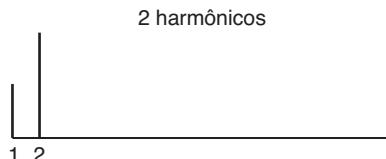
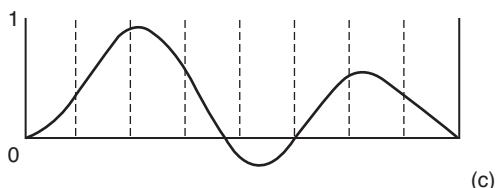
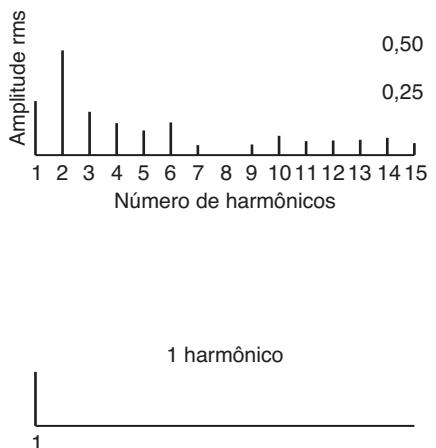
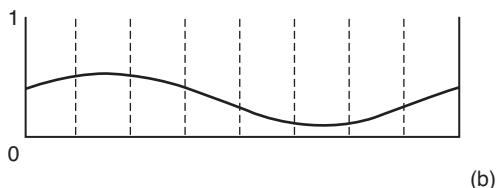



Figura 2.1 (a) Um sinal binário e a raiz quadrada média das amplitudes de Fourier. (b)-(e) Aproximações sucessivas do sinal original.

A raiz quadrada média das amplitudes, $\sqrt{a_n^2 + b_n^2}$ para os primeiros termos são mostradas no lado direito da Figura 2.1(a). Esses valores são importantes porque seus quadrados são proporcionais à energia transmitida na frequência correspondente.

Nenhum recurso de transmissão é capaz de transmitir sinais sem perder parte da energia no processo. Se todos os coeficientes da série de Fourier fossem igualmente reduzidos, o sinal resultante seria reduzido em amplitude, mas não distorcido [isto é, ele teria a mesma forma mostrada na Figura 2.1(a)]. Infelizmente, todos os meios de transmissão reduzem diferentes componentes de Fourier por diferentes valores e, em consequência disso, introduzem distorção. Em geral, para um fio, as amplitudes são transmitidas sem redução, de 0 até alguma frequência f_c [medida em ciclos/s ou Hertz (Hz)], com todas as frequências acima dessa frequência de corte (*cutoff*) sendo atenuadas. A faixa de frequências transmitidas sem serem fortemente atenuadas denomina-se **largura de banda**. Na prática, o corte não é nítido; assim, muitas vezes a largura de banda varia desde 0 até a frequência em que a potência recebida caiu para a metade.

A largura de banda é uma propriedade física do meio de transmissão, que depende, por exemplo, da construção, da espessura e do comprimento do meio (fio ou fibra). Os filtros normalmente são usados para limitar ainda mais o volume de largura de banda de um sinal. Os canais sem fio 802.11 têm permissão para usar até aproximadamente 20 MHz, por exemplo, de modo que rádios 802.11 filtram a largura de banda do sinal para essa faixa. Como outro exemplo, os canais de televisão tradicionais (analógicos) ocupam 6 MHz cada, em um fio ou pelo ar. Essa filtragem permite que mais sinais compartilhem determinada região do espectro, o que melhora a eficiência geral do sistema. Isso significa que a faixa de frequência para alguns sinais não começará em zero, mas isso não importa. A largura de banda ainda é a largura da banda de frequências que são passadas, e a informação que pode ser transportada depende apenas dessa largura, e não das frequências inicial e final. Os sinais que vão de 0 para cima, até uma frequência máxima, são chamados sinais de **banda base**. Sinais que são deslocados para ocupar uma faixa de frequências mais alta, como acontece para todas as transmissões sem fio, são chamados sinais de **banda passante**.

Vejamos agora como seria a forma do sinal da Figura 2.1(a) se a largura de banda fosse tão estreita que apenas as frequências mais baixas fossem transmitidas (ou seja, se a função estivesse sendo aproximada pelos primeiros termos da Equação 2.1). A Figura 2.1(b) mostra o sinal resultante de um canal pelo qual apenas o primeiro harmônico (o fundamental, f) pode passar. Da mesma forma, a Figura 2.1(c)-(e) mostra os espectros e as funções reconstruídas para canais com uma largura de banda mais alta. Para a transmissão digital, o objetivo é receber um sinal com fidelidade suficiente para reconstruir a sequência de bits que foi enviada. Já podemos fazer isso com facilidade na Figura 2.1(e), de modo que é um desperdício usar mais harmônicos para receber uma réplica mais precisa.

Dada uma taxa de bits de b bits/s, o tempo necessário para o envio de 8 bits (em nosso exemplo, um bit de cada vez) é $8/b$ s; assim, a frequência do primeiro harmônico desse sinal é $b/8$ Hz. Uma linha telefônica comum, frequentemente chamada **linha de qualidade de voz**, tem uma frequência de corte artificialmente introduzida, pouco acima de 3.000 Hz. A presença dessa restrição significa que o número do harmônico mais alto transmitido é aproximadamente $3.000/(b/8)$ ou $24.000/b$ (o corte não é preciso).

Para algumas taxas de dados, os números funcionam de acordo com o padrão mostrado na Tabela 2.1. Esses números deixam claro que, quando se tenta fazer uma transmissão a 9.600 bps usando uma linha telefônica de qualidade de voz, o modelo sugerido na Figura 2.1(a) assume a forma mostrada na Figura 2.1(c), tornando complicada a recepção precisa do fluxo original de bits. Podemos perceber também que, em taxas de dados bem mais altas que 38,4 kbps, não existe a menor possibilidade de que todos os sinais sejam *binários*, mesmo quando não há o menor ruído no meio de transmissão. Em outras palavras, limitando-se a largura de banda, limita-se a taxa de dados, mesmo nos canais perfeitos (sem ruídos). No entanto, sofisticados esquemas de codificação que usam diversos níveis de tensão possibilitam a existência e a utilização de taxas de dados mais altas. Vamos discutir essa questão ao longo deste capítulo.

Há muita confusão sobre largura de banda, pois ela significa coisas diferentes para engenheiros elétricos e para cientistas da computação. Para os engenheiros elétricos, a largura de banda (analógica) é, como descrevemos, uma quantidade medida em Hz. Para os cientistas da computação, a largura de banda (digital) é a taxa de dados máxima de um canal, uma quantidade medida em bits/s. Essa taxa de dados é o resultado final do uso da largura de banda

Bps	T (ms)	Primeiro harmônico (Hz)	Número de harmônicos enviados
300	26,67	37,5	80
600	13,33	75	40
1.200	6,67	150	20
2.400	3,33	300	10
4.800	1,67	600	5
9.600	0,83	1.200	2
19.200	0,42	2.400	1
38.400	0,21	4.800	0

Tabela 2.1 | Relação entre as taxas de dados e os harmônicos em nosso exemplo.

analógica de um canal físico para a transmissão digital, e as duas estão relacionadas, conforme veremos a seguir. Neste livro, o contexto esclarecerá se queremos dizer largura de banda analógica (Hz) ou largura de banda digital (bits/s).

2.1.3 Taxa máxima de dados de um canal

Em 1924, Henry Nyquist, um engenheiro da AT&T, percebeu que até mesmo um canal perfeito tem uma capacidade de transmissão finita. Ele derivou uma equação expressando a taxa máxima de dados de um canal sem ruído com largura de banda finita. Em 1948, Claude Shannon aprofundou o trabalho de Nyquist e o estendeu ao caso de um canal sujeito a ruído aleatório (isto é, ruído termodinâmico) (Shannon, 1948). Esse artigo é o mais importante de toda a teoria da informação. Veja a seguir um resumo dos resultados, agora clássicos.

Nyquist provou que, se um sinal arbitrário atravessar um filtro passa-baixa de largura de banda B, o sinal filtrado poderá ser completamente reconstruído a partir de apenas $2B$ amostras (exatas) por segundo. Fazer uma amostragem da linha com uma rapidez maior que $2B$ vezes por segundo seria inútil, pois os componentes de frequência mais alta que essa amostragem poderia recuperar já teriam sido filtrados. Se o sinal consistir em V níveis discretos, o teorema de Nyquist afirma que:

$$\text{taxa máxima de dados} = 2B \log_2 V \text{ bits/s} \qquad (2.2)$$

Por exemplo, um canal de 3 kHz sem ruído não pode transmitir sinais binários (ou seja, de dois níveis) a uma taxa superior a 6.000 bps.

Até agora, só mencionamos os canais sem ruído. Se houver ruído aleatório, a situação se deteriorará com rapidez. Além disso, sempre existe ruído aleatório (térmico) presente, em virtude do movimento das moléculas no sistema. O volume de ruído térmico presente é medido pela relação entre a potência do sinal e a do ruído, chamada relação sinal/ruído, ou **SNR** (**Signal-to-Noise Ratio**). Se representarmos a potência do sinal por S e a potência do ruído por N, a relação sinal/ruído será S/N. Em geral, não se faz referência à relação propriamente dita; em vez disso, utiliza-se a quantidade $10 \log_{10} S/N$, pois ela pode variar por uma faixa muito grande. As unidades da escala logarítmica são chamadas **decibéis** (**dB**), com 'deci' significando 10 e 'bel' escolhido em honra a Alexander Graham Bell, que inventou o telefone. Uma relação S/N igual a 10 corresponde a 10 dB, uma relação igual a 100 equivale a 20 dB, uma relação igual a 1.000 equivale a 30 dB e assim por diante. Com frequência, os fabricantes de amplificadores estereofônicos caracterizam a largura de banda (faixa de frequência) na qual seu produto é linear oferecendo a frequência de 3 dB em cada extremidade. Esses são os pontos em que o fator de amplificação foi dividido aproximadamente ao meio (porque $10 \log_{10} 0{,}5 \approx -3$).

O principal resultado de Shannon é que a taxa máxima de dados ou **capacidade** de um canal com ruídos cuja largura de banda é B Hz, e cuja relação sinal/ruído é S/N, é dada por:

$$\text{número máximo de bits/s} = B \log_2 (1 + S/N) \qquad (2.3)$$

Isso nos diz as melhores capacidades que os canais reais podem ter. Por exemplo, ADSL (Asymmetric Digital Subscriber Line), que oferece acesso à Internet por linhas telefônicas normais, usa uma largura de banda com cerca de 1 MHz. A SNR depende muito da distância da central telefônica até a casa, e uma SNR com cerca de 40 dB para linhas curtas de 1 a 2 km é muito boa. Com essas características, o canal nunca pode transmitir muito mais do que 13 Mbps, não importa quantos níveis de sinal sejam usados e não importa com que frequência as amostras são tomadas. Na prática, a ADSL é especificada para até 12 Mbps, embora os usuários normalmente vejam taxas mais baixas. Essa taxa de dados é realmente muito boa, com mais de 60 anos de técnicas de comunicação tendo reduzido bastante a lacuna entre a capacidade de Shannon e a capacidade dos sistemas reais.

O resultado de Shannon utilizou os argumentos da teoria da informação e se aplica a qualquer canal sujeito a ruído térmico. Os exemplos contrários devem ser tratados na mesma categoria das máquinas de movimento contínuo (ou moto-perpétuo). Para a ADSL ultrapassar os 13 Mbps, ela deve melhorar a SNR (por exemplo, inserindo repetidores digitais nas linhas mais próximas do cliente) ou usar mais largura de banda, como é feito com a evolução para ADSL2+.

2.2 Meios de transmissão guiados

O objetivo da camada física é transmitir um fluxo bruto de bits de uma máquina para outra. Vários meios físicos podem ser usados para realizar a transmissão real. Cada um tem seu próprio nicho em termos de largura de banda, atraso, custo e facilidade de instalação e manutenção. Os meios físicos são agrupados em meios guiados, como fios de cobre e fibras ópticas, e em meios não guiados, como as redes terrestres sem fios, satélite e os raios laser transmitidos pelo ar. Discutiremos os meios de transmissão guiados nesta seção e os meios não guiados, nas próximas seções.

2.2.1 Meios magnéticos

Uma das formas mais comuns de transportar dados de um computador para outro é gravá-los em fita magnética ou em mídia removível (por exemplo, DVDs graváveis) e transportar fisicamente a fita ou os discos para a máquina de destino, onde eles finalmente serão lidos. Apesar de não ser tão sofisticado quanto a utilização de um satélite de comunicação geossíncrono, esse método costuma ser muito

mais eficaz sob o ponto de vista financeiro, em especial nas aplicações em que a alta largura de banda ou o custo por bit têm importância fundamental.

Um cálculo simples esclarecerá essa questão. Uma fita Ultrium de padrão industrial pode armazenar 800 gigabytes. Uma caixa de 60 × 60 × 60 cm pode conter cerca de 1.000 fitas desse tipo, perfazendo uma capacidade total de 800 terabytes, ou 6.400 terabits (6,4 petabits). Uma caixa de fitas pode ser entregue em qualquer parte do país em 24 horas pelo serviço de Sedex dos Correios, pela Federal Express e por outras transportadoras. A largura de banda efetiva dessa transmissão é de 6.400 terabits/86.400 s, ou um pouco mais de 70 Gbps. Se o destino estiver a apenas uma hora de distância, a largura de banda será ampliada para mais de 1.700 Gbps. Nenhuma rede de computadores consegue nem mesmo se aproximar desse desempenho. Logicamente, as redes estão ficando mais rápidas, mas as densidades das fitas também estão aumentando.

Se considerarmos o custo, obteremos um quadro semelhante. O custo de uma fita Ultrium é de aproximadamente US$ 40 quando a compra é feita no atacado. Uma fita pode ser reutilizada pelo menos dez vezes. Portanto, o custo das fitas passa a ser US$ 4.000 por caixa, para cada utilização. Adicione a esse montante mais US$ 1.000 pelo frete (provavelmente muito menos) e teremos um custo final aproximado de US$ 5.000 para transportar 800 TB. Consequentemente, para transportar 1 gigabyte, gastaremos pouco mais de meio centavo de dólar. Nenhuma rede pode competir com esses valores. Moral da história:

Nunca subestime a largura de banda de uma caminhonete cheia de fitas 'voando' na estrada.

2.2.2 Pares trançados

Embora as características de largura de banda da fita magnética sejam excelentes, as características de atraso são ruins. O tempo de transmissão é medido em minutos ou horas, e não em milissegundos. Muitas aplicações precisam de uma conexão on-line. Um dos meios de transmissão mais antigos e ainda mais comuns é o **par trançado**. Um par trançado consiste em dois fios de cobre encapados, que em geral tem cerca de 1 mm de espessura. Os fios são enrolados de forma helicoidal, assim como uma molécula de DNA. O trançado dos fios é feito porque dois fios paralelos formam uma antena simples. Quando os fios são trançados, as ondas de diferentes partes dos fios se cancelam, o que significa menor interferência. Um sinal normalmente é transportado a partir da diferença das tensões terminais (diferença de potencial – ddp) entre os dois fios no par. Isso oferece melhor imunidade ao ruído externo, pois o ruído tende a afetar os dois fios da mesma forma, mantendo a ddp inalterada.

A aplicação mais comum do par trançado é o sistema telefônico. Quase todos os telefones estão conectados à estação central da companhia telefônica por um par trançado. Tanto as chamadas telefônicas quanto o acesso à Internet por ADSL utilizam essas linhas. Os pares trançados podem se estender por diversos quilômetros sem amplificação, mas, quando se trata de distâncias mais longas, o sinal é atenuado e existe a necessidade de repetidores. Quando muitos pares trançados percorrem paralelamente uma distância muito grande, como acontece na ligação entre um prédio e a estação central da companhia telefônica, eles são envolvidos por uma capa protetora. Se não estivessem trançados, esses pares provocariam muitas interferências. Em locais onde as linhas telefônicas são instaladas em postes, é comum vermos cabos de pares trançados com vários centímetros de diâmetro.

Os pares trançados podem ser usados na transmissão de sinais analógicos ou digitais. A largura de banda depende da espessura do fio e da distância percorrida, mas, em muitos casos, é possível alcançar diversos megabits/s por alguns quilômetros. Em virtude do custo e do desempenho obtidos, os pares trançados são usados em larga escala e é provável que permaneçam assim nos próximos anos.

O cabeamento de par trançado pode ser de vários tipos. A variedade mais comum empregada em muitos prédios de escritórios é chamada cabeamento de **Categoria 5**, ou 'Cat 5'. Um par trançado de categoria 5 consiste em dois fios isolados e levemente trançados. Quatro pares desse tipo normalmente são agrupados em uma capa plástica para proteger os fios e mantê-los juntos. Esse arranjo pode ser visto na Figura 2.2.

Diferentes padrões de LAN podem usar os pares trançados de formas diferentes. Por exemplo, a Ethernet de 100 Mbps usa dois (dos quatro) pares, um para cada direção. Para alcançar velocidades mais altas, a Ethernet de 1 Gbps usa todos os quatro pares nas duas direções simultaneamente; isso requer que o receptor decomponha o sinal que é transmitido localmente.

Neste ponto, devemos explicar alguma terminologia geral. Os enlaces que podem ser usados nos dois sentidos ao mesmo tempo, como uma estrada de mão dupla, são chamados enlaces **full-duplex**. Ao contrário, os que são usados em qualquer sentido, mas apenas um deles de cada vez, como uma linha férrea de trilho único, são chamados enlaces **half-duplex**. Uma terceira categoria consiste em enlaces que permitem o tráfego em apenas uma direção,

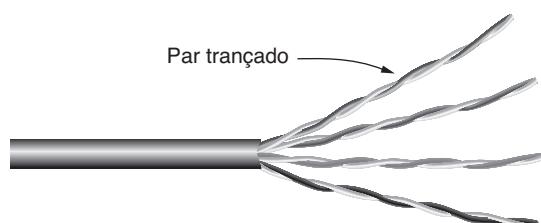



Figura 2.2 | Cabo UTP Categoria 5 com quatro pares trançados.

como uma rua de mão única. Eles são chamados enlaces **simplex**.

Retornando ao par trançado, os cabos Cat 5 substituíram os cabos **Categoria 3** mais antigos com um cabo semelhante que usa o mesmo conector, mas com mais voltas por metro. Mais voltas resultam em menos interferências e em um sinal de melhor qualidade por distâncias maiores, tornando os cabos mais adequados para a comunicação de computador de alta velocidade, especialmente LANs Ethernet de 100 Mbps e 1 Gbps.

Os fios mais novos provavelmente serão de **Categoria 6** ou mesmo de **Categoria 7**. Essas categorias possuem especificações mais rígidas para lidar com sinais de larguras de banda maiores. Alguns cabos na Categoria 6 e superiores são usados para sinais de 500 MHz e podem aceitar enlaces de 10 Gbps, que logo serão implementados.

Até a Categoria 6, esses tipos de fios são conhecidos como **par trançado não blindado**, ou **UTP** (**Unshielded Twisted Pair**), pois consistem simplesmente em fios e isolamento. Ao contrário, os cabos de Categoria 7 possuem uma blindagem nos pares de fios individuais e também ao redor do cabo inteiro (dentro da capa plástica protetora). A blindagem reduz a suscetibilidade à interferência externa e linha cruzada com outros cabos vizinhos, atendendo às especificações de desempenho exigidas. Os cabos são reminiscências dos cabos de pares trançados blindados de alta qualidade, porém grossos e caros, que a IBM introduziu no início da década de 1980, mas que não se tornaram populares fora das instalações da empresa. Evidentemente, é hora de tentar novamente.

2.2.3 Cabo coaxial

Outro meio de transmissão comum é o **cabo coaxial** (conhecido por muitos apenas como 'coax'). Ele tem melhor blindagem que os pares trançados e, assim, pode se estender por distâncias mais longas em velocidades mais altas. Dois tipos de cabo coaxial são amplamente utilizados. Um deles, o cabo de 50 ohms, é comumente empregado nas transmissões digitais. O outro tipo, o cabo de 75 ohms, é usado com frequência nas transmissões analógicas e de televisão a cabo. Essa distinção se baseia mais em fatores históricos do que técnicos (por exemplo, as primeiras antenas dipolo tinham uma impedância de 300 ohms e era fácil desenvolver transformadores de casamento de impedância de 4:1). Começando em meados da década de 1990, as operadoras de TV a cabo começaram a oferecer acesso à Internet por cabo, o que tornou o cabo de 75 ohms mais importante para a comunicação de dados.

Um cabo coaxial consiste em um fio de cobre esticado na parte central, protegido por um material isolante. O isolante é envolvido por um condutor cilíndrico, geralmente como uma malha sólida entrelaçada. O condutor externo é coberto por uma camada plástica protetora. A Figura 2.3 apresenta uma vista de corte de um cabo coaxial.

A construção e a blindagem do cabo coaxial proporcionam a ele uma boa combinação de alta largura de banda e excelente imunidade ao ruído. A largura de banda possível depende da qualidade e do tamanho do cabo. Os cabos modernos têm uma largura de banda de até alguns GHz. Os cabos coaxiais eram muito usados no sistema telefônico para linhas de longa distância, mas agora estão sendo substituídos por fibras ópticas nas rotas de longa distância. Porém, os cabos coaxiais ainda são usados em larga escala pelas redes de televisão a cabo e em redes metropolitanas.

2.2.4 Linhas de energia elétrica

As redes de telefonia e de televisão a cabo não são as únicas fontes de fiação que podem ser reutilizadas para a comunicação de dados. Há um outro tipo de fiação ainda mais comum: as linhas de energia elétrica. Estas oferecem energia elétrica às casas, e a fiação elétrica dentro das casas distribui a potência às tomadas elétricas.

O uso das linhas de energia elétrica para comunicação de dados é uma ideia antiga. Essas têm sido usadas pelas companhias de eletricidade para a comunicação de baixo nível, como a medição remota, há muitos anos, bem como para controlar dispositivos em casa (por exemplo, o padrão X10). Nos últimos anos, tem havido um interesse renovado na comunicação de alto nível por essas linhas, tanto dentro de casa, como uma LAN, quanto fora dela, para o acesso de banda larga à Internet. Vamos nos concentrar no cenário mais comum: usar fios elétricos dentro da casa.

A conveniência de usar linhas de energia para a rede deve ser clara. Basta conectar uma TV e um receptor na parede, o que você precisa fazer de qualquer forma, pois ele precisa de energia, e ele poderá enviar e receber filmes pela fiação elétrica. Essa configuração pode ser vista na Figura 2.4. Não há outro conector ou rádio. O sinal de

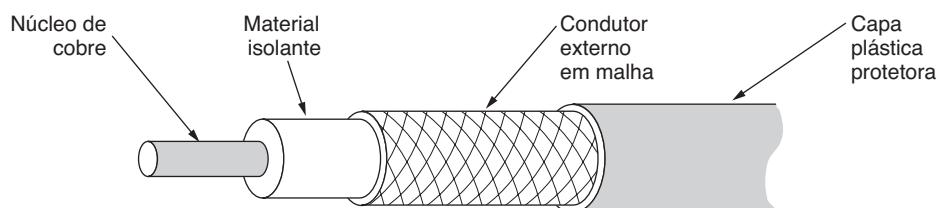



Figura 2.3 | Um cabo coaxial.

dados é sobreposto ao sinal de baixa frequência (ou no fio ativo, ou 'quente'), enquanto os dois sinais usam a fiação ao mesmo tempo.

A dificuldade em usar a fiação elétrica domiciliar como uma rede é que ela foi projetada para distribuir energia elétrica. Essa tarefa é muito diferente de distribuir sinais de dados, algo para o qual a fiação doméstica é pouco eficiente. Os sinais elétricos são enviados a 50-60 Hz e a fiação atenua os sinais de frequência muito mais alta (MHz) necessários para a comunicação de dados de alto nível. As propriedades elétricas da fiação variam de uma casa para outra e mudam à medida que os aparelhos são ligados e desligados, fazendo com que os sinais de dados oscilem pela fiação. As correntes transitórias quando os aparelhos são ligados e desligados criam ruído por uma larga faixa de frequências. Sem o trançado cuidadoso dos pares trançados, a fiação elétrica atua como uma boa antena, apanhando sinais externos e emitindo sinais próprios. Esse comportamento significa que, para atender aos requisitos da regulamentação, o sinal de dados precisa excluir frequências licenciadas, como as faixas de radioamador.

Apesar dessas dificuldades, é possível enviar pelo menos 100 Mbps pela fiação elétrica doméstica usando esquemas de comunicação que resistem a frequências enfraquecidas e eclosões de erros. Muitos produtos usam diversos padrões próprios para as redes de energia elétrica, de modo que os padrões internacionais estão em desenvolvimento ativo.

2.2.5 Fibra óptica

Muitas pessoas na indústria de informática se orgulham da rapidez com que a tecnologia usada nos computadores vem melhorando, conforme a lei de Moore, que prevê a duplicação do número de transistores por chip a cada dois anos aproximadamente (Schaller, 1997). O IBM PC original (de 1981) funcionava com uma velocidade de clock de 4,77 MHz. Vinte e oito anos depois, os PCs podiam usar uma CPU de quatro núcleos a 3 GHz. Esse aumento é um ganho de fator em torno de 2.500, ou 16 vezes por década. Impressionante.

No mesmo período, enlaces de comunicação remotos passaram de 45 Mbps (uma linha T3 no sistema telefônico) para 100 Gbps (uma linha moderna de longa distância). Esse ganho também é impressionante, um fator de mais de 2.000 e perto de 16 vezes em cada década, enquanto, no mesmo período, a taxa de erros passou de 10^{-5} por bit para quase zero. Além disso, as CPUs isoladas estão se aproximando dos limites físicos, motivo pelo qual agora é o número de CPUs que está sendo aumentado por chip. Por outro lado, com a atual tecnologia de fibra óptica, a largura de banda pode ultrapassar a casa dos 50.000 Gbps (50 Tbps) e nem estamos perto de alcançar esses limites. O limite prático atual é de cerca de 100 Gbps, em razão de nossa incapacidade de realizar a conversão entre sinais elétricos e ópticos em uma velocidade maior. Para criar enlaces de maior capacidade, muitos canais correm em paralelo por uma única fibra.

Nesta seção, estudaremos a fibra óptica para descobrir como funciona essa tecnologia de transmissão. Na corrida entre informática e comunicação, esta pode ganhar em virtude das redes de fibra óptica. As implicações reais disso seriam essencialmente largura de banda infinita e uma nova premissa de que todos os computadores são terrivelmente lentos e, por essa razão, as redes deveriam tentar evitar a computação a todo custo, independentemente do desperdício de largura de banda que isso signifique. Essa mudança levará algum tempo para entrar na cabeça de uma geração de cientistas da computação e engenheiros ensinados a pensar em termos dos baixos limites de Shannon, impostos pelo cobre.

Naturalmente, esse cenário não diz tudo, pois não inclui custos. O custo para instalar fibra até a última milha e chegar aos consumidores, evitando a baixa largura de banda dos fios e a disponibilidade limitada de espectro, é muito alto. Também custa mais energia para mover os bits do que para a computação. Sempre podemos ter ilhas de injustiça, onde a computação ou a comunicação é basicamente gratuita. Por exemplo, na borda da Internet, usamos computação e armazenamento para o problema de compactação e caching de conteúdo, tudo para fazer melhor uso dos enlaces de acesso à Internet. Dentro da Internet, podemos fazer o contrário, com empresas como a Google movendo grandes quantidades de dados pela rede até onde for mais barato armazená-los ou computá-los.

A fibra óptica é usada para transmissão por longa distância nos backbones da rede, LANs de alta velocidade (embora, até aqui, o cobre sempre tenha conseguido acompanhar) e acesso à Internet em alta velocidade, como

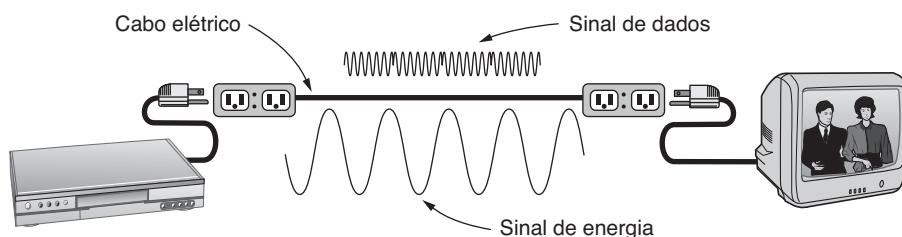



Figura 2.4 | Rede de dados que usa a fiação elétrica domiciliar.

FTTH (**Fiber to the Home**). Um sistema de transmissão óptico tem três componentes-chave: a fonte de luz, o meio de transmissão e o detector. Convencionalmente, um pulso de luz indica um bit 1 e a ausência de luz indica um bit 0. O meio de transmissão é uma fibra de vidro ultrafina. O detector gera um pulso elétrico quando a luz incide sobre ele. Conectando uma fonte de luz em uma ponta de uma fibra óptica e um detector na outra, temos um sistema de transmissão de dados unidirecional que aceita um sinal elétrico, o converte e o transmite por pulsos de luz e depois novamente converte a saída para um sinal elétrico na ponta receptora.

Esse sistema de transmissão perderia luz para o meio e seria inútil na prática se não fosse por um princípio interessante da física. Quando um raio de luz passa de um meio para outro — por exemplo, de sílica fundida para o ar —, o raio é refratado (inclinado) no limite sílica/ar, como mostra a Figura 2.5(a). Aqui, vemos um raio de luz incidindo no limite em um ângulo α_1 emergindo em um ângulo β_1. A quantidade de refração depende das propriedades das duas mídias (em particular, seus índices de refração). Para ângulos de incidência acima de um certo valor crítico, a luz é refratada de volta para a sílica; nada escapa para o ar. Assim, um raio de luz incidente em um ângulo crítico ou acima é interceptado dentro da fibra, como mostra a Figura 2.5(b), e pode se propagar por muitos quilômetros praticamente sem perdas.

O exemplo da Figura 2.5(b) mostra apenas um raio interceptado, mas, como qualquer raio de luz incidente na fronteira acima do ângulo crítico será refletido internamente, muitos raios distintos estarão ricocheteando em diferentes ângulos. Dizemos que cada raio tem um modo específico; assim, uma fibra que apresenta essa propriedade é chamada **fibra multimodo**.

No entanto, se o diâmetro da fibra for reduzido a alguns comprimentos de onda de luz, a fibra agirá como um guia de onda e a luz só poderá se propagar em linha reta, sem ricochetear, produzindo, assim, uma **fibra de modo único** ou **fibra monomodo**. As fibras de modo único são mais caras, mas são amplamente utilizadas em distâncias mais longas. As fibras de modo único disponíveis no momento podem transmitir dados a 100 Gbps por 100 km sem amplificação. Foram obtidas taxas de dados ainda mais altas em laboratório, para distâncias mais curtas.

Transmissão de luz na fibra

As fibras ópticas são feitas de vidro, que, por sua vez, é produzido a partir da areia, uma matéria-prima de baixo custo e abundante. Os antigos egípcios já dominavam a manufatura do vidro, mas o vidro produzido por eles não podia ter mais de 1 mm de espessura para que a luz pudesse atravessá-lo. O vidro transparente usado nas janelas foi desenvolvido durante a Renascença. O vidro usado nas modernas fibras ópticas é tão transparente que se, em vez de água, os oceanos fossem cheios desse tipo de vidro, seria possível ver o fundo do mar da superfície, da mesma forma que é possível ver o solo quando voamos de avião em um dia claro.

A atenuação de luz através do vidro depende do comprimento de onda da luz (bem como de algumas propriedades físicas do vidro). Ela é definida como a razão da potência do sinal de entrada e saída. Para o tipo de vidro usado nas fibras, a atenuação é mostrada na Figura 2.6, em decibéis por quilômetro linear de fibra. Por exemplo, quando o fator de perda é igual a dois, obtemos uma atenuação de $10 \log_{10} 2 = 3$ dB. A figura mostra a parte do infravermelho do espectro que, na prática, é a utilizada. A luz visível tem comprimentos de onda ligeiramente mais curtos, que variam de 0,4 a 0,7 mícron (1 mícron é igual a 10^{-6} metros). Na verdade, esses comprimentos de onda seriam de 400 nm a 700 nm, mas manteremos a nomenclatura tradicional.

A comunicação óptica comumente utiliza três bandas de comprimentos de onda. Elas são centralizadas em 0,85, 1,30 e 1,55 micra, respectivamente. As três bandas têm entre 25.000 e 30.000 GHz de largura. A banda de 0,85 mícron foi usada primeiro. Ela tem maior atenuação e, por isso, é usada para distâncias mais curtas, mas, nesse comprimento de onda, os lasers e os circuitos eletrônicos podem ser produzidos a partir do mesmo material (arseneto de gálio). As duas últimas têm boas propriedades de atenuação (uma perda inferior a 5 por cento por quilômetro). A banda de 1,55 mícron agora é muito utilizada com amplificadores dopados com érbio, que funcionam diretamente no domínio óptico.

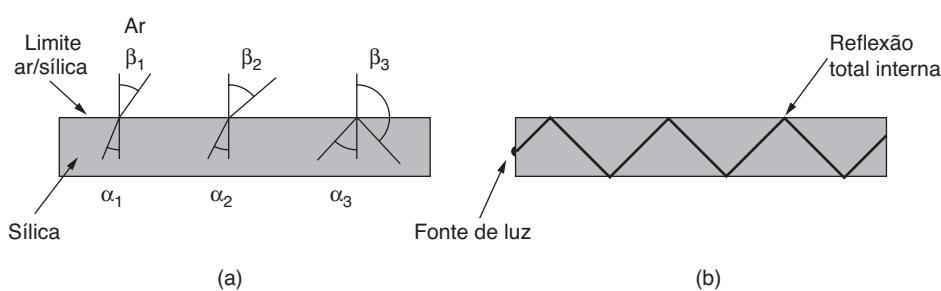



Figura 2.5 | (a) Três exemplos de um raio de luz dentro de uma fibra de sílica incidindo na fronteira ar/sílica em diferentes ângulos. (b) A luz interceptada pela reflexão total interna.

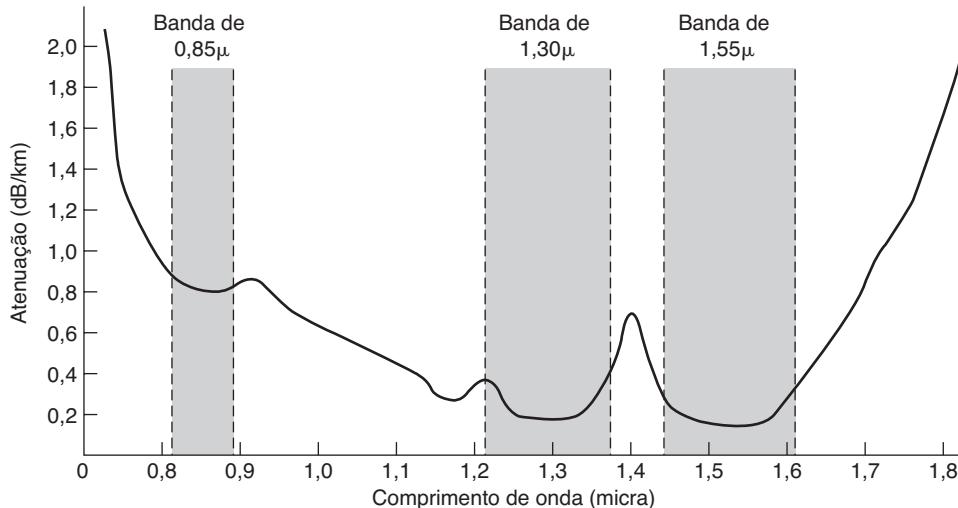



Figura 2.6 | Atenuação da luz na fibra, na região do infravermelho.

Os pulsos de luz enviados através de uma fibra se expandem à medida que se propagam. Essa expansão é chamada **dispersão cromática**. O volume da dispersão depende do comprimento de onda. Uma forma de impedir que esses pulsos dispersos se sobreponham é aumentar a distância entre eles, mas isso só pode ser feito reduzindo-se a taxa de sinalização. Felizmente, descobriu-se que, quando os pulsos são produzidos em uma forma especial relacionada ao recíproco do cosseno hiperbólico, praticamente todos os efeitos de dispersão são cancelados e é possível enviar pulsos por milhares de quilômetros sem que haja uma distorção significativa. Esses pulsos são chamados **sólitons**. Atualmente, o mundo assiste a um grande esforço de pesquisa voltado para colocar em prática as experiências que estão sendo feitas em laboratórios com os sólitons.

Cabos de fibra

Os cabos de fibra óptica são semelhantes aos cabos coaxiais, exceto por não terem a malha metálica. A Figura 2.7(a) mostra a vista lateral de uma única fibra. No centro fica o núcleo de vidro através do qual a luz se propaga. Nas fibras multimodo, o núcleo normalmente tem 50 micra de diâmetro, o que corresponde à espessura de um fio de cabelo humano. Nas fibras de modo único, o núcleo tem entre 8 e 10 micra.

O núcleo é envolvido por um revestimento de vidro com um índice de refração inferior ao do núcleo, para manter toda a luz nele. Em seguida, há uma cobertura de plástico fino para proteger o revestimento interno. Geralmente, as fibras são agrupadas em feixes, protegidas por um revestimento externo. A Figura 2.7(b) mostra um cabo com três fibras.

Normalmente, os cabos de fibra terrestres são colocados no solo a um metro da superfície, onde ocasionalmente são atacados por retroescavadeiras ou roedores. Próximo ao litoral, cabos de fibra transoceânicos são enterrados em trincheiras por uma espécie de arado marítimo. Em águas profundas, eles são depositados no fundo, onde podem ser arrastados por redes de pesca ou comidos por lulas gigantes.

As fibras podem estar conectadas de três maneiras diferentes. Primeiro, elas podem ter conectores em suas extremidades e serem plugadas em soquetes de fibra. Os conectores perdem de 10 a 20 por cento da luz, mas facilitam a reconfiguração dos sistemas.

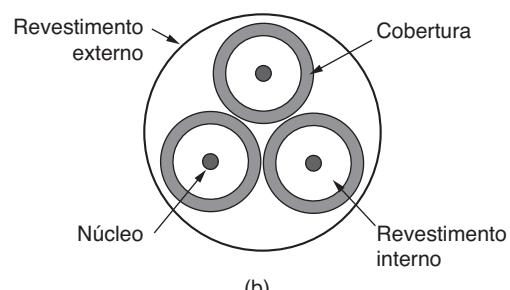
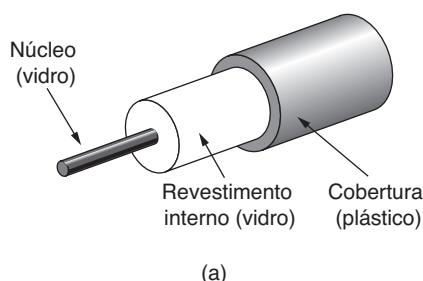



Figura 2.7 | (a) Vista lateral de uma única fibra. (b) Vista da extremidade de um cabo com três fibras.

Segundo, elas podem ser unidas mecanicamente. Nesse caso, as duas extremidades são cuidadosamente colocadas uma perto da outra em uma luva especial e fixadas no lugar. O alinhamento pode ser melhorado fazendo-se a luz passar pela junção e, em seguida, realizando-se pequenos ajustes cuja finalidade é maximizar o sinal. As junções mecânicas são realizadas em cerca de cinco minutos por uma equipe treinada e resultam em uma perda de 10 por cento da luz.

Terceiro, duas peças de fibra podem ser fundidas de modo a formar uma conexão sólida. A união por fusão é quase tão boa quanto uma fibra sem emendas; no entanto, mesmo nesse caso, há uma pequena atenuação.

Nos três tipos de uniões podem ocorrer reflexões no ponto de junção, e a energia refletida pode interferir no sinal.

Dois tipos de fontes de luz geralmente são usados para fazer a sinalização: os diodos emissores de luz (*light emitting diodes* — LEDs) e os lasers semicondutores. Eles têm diferentes propriedades, como mostra a Tabela 2.2. O comprimento de onda desses elementos pode ser ajustado pela inserção de interferômetros de Fabry-Perot ou Mach-Zehnder entre a fonte e a fibra. Os interferômetros de Fabry-Perot são cavidades ressonantes simples que consistem em dois espelhos paralelos. A luz incide perpendicularmente aos espelhos. O comprimento da cavidade filtra os comprimentos de onda que cabem em um número inteiro de períodos. Os interferômetros de Mach-Zehnder separam a luz em dois feixes. Estes percorrem distâncias ligeiramente diferentes. Eles são recombinados no destino e só ficam em fase para certos comprimentos de onda.

A extremidade de recepção de uma fibra óptica consiste em um fotodiodo, que emite um pulso elétrico ao ser atingido pela luz. O tempo de resposta de um fotodiodo, que converte o sinal do domínio óptico para o elétrico, limita as taxas de dados a cerca de 100 Gbps. O ruído térmico também é importante, pois um pulso de luz deve conduzir energia suficiente para ser detectado. Com pulsos de potência suficiente, a taxa de erros pode se tornar arbitrariamente pequena.

Comparação entre fibras ópticas e fios de cobre

É instrutivo comparar a fibra com o cobre. A fibra tem muitas vantagens. Para começar, ela pode gerenciar larguras de banda muito mais altas do que o cobre. Essa característica sozinha já justificaria seu uso nas redes de última geração. Em razão da baixa atenuação, os repetidores só são necessários a cada 50 quilômetros de distância em linhas longas, em comparação com a distância de 5 km no caso do cobre, resultando em uma economia de custo significativa. A fibra também tem a vantagem de não ser afetada por picos de tensão, interferência eletromagnética ou quedas no fornecimento de energia. Ela também está imune à ação corrosiva de alguns elementos químicos que pairam no ar, o que é importante em ambientes industriais desfavoráveis.

Por mais estranho que possa parecer, as empresas telefônicas gostam da fibra por outra razão: ela é fina e leve. Muitos dos dutos de cabos atuais estão completamente lotados, de modo que não há espaço para aumentar sua capacidade. Além da remoção e subsequente substituição de todo o cobre por fibras esvaziar os dutos, o cobre tem um excelente valor de revenda para as refinarias especializadas, pois trata-se de um minério de altíssima qualidade. Além disso, a fibra é muito mais leve que o cobre. Mil pares trançados com 1 km de comprimento pesam 8 toneladas. Duas fibras têm mais capacidade e pesam apenas 100 kg, reduzindo de maneira significativa a necessidade de sistemas mecânicos de suporte, que exigem mais manutenção. Nas novas rotas, as fibras são preferidas por terem um custo de instalação muito mais baixo. Por fim, as fibras não desperdiçam luz e dificilmente são interceptadas. Por essas razões, a fibra é uma alternativa com excelente nível de segurança contra possíveis escutas telefônicas.

No entanto, a fibra tem a desvantagem de ser uma tecnologia menos familiar, exigindo conhecimentos que nem todos os engenheiros possuem e, além disso, as fibras podem ser danificadas com facilidade, se forem encurvadas demais. Como a transmissão óptica é basicamente unidirecional, a comunicação bidirecional exige duas fibras ou duas bandas de frequência em uma única fibra. Por

Item	LED	Laser semicondutor
Taxa de dados	Baixa	Alta
Tipo de fibra	Multimodo	Multimodo ou modo único
Distância	Curta	Longa
Vida útil	Longa	Curta
Sensibilidade à temperatura	Insignificante	Substancial
Custo	Baixo	Dispendioso

Tabela 2.2 | Uma comparação entre laser semicondutor e LEDs utilizados como fontes de luz.

fim, as interfaces de fibra são mais caras que as interfaces elétricas. Apesar disso, o futuro de toda a comunicação fixa de dados para distâncias superiores a alguns metros depende claramente da fibra. Para obter mais informações sobre todos os aspectos das fibras ópticas e de suas redes, consulte Hecht (2005).

2.3 Transmissão sem fios

Estamos assistindo ao surgimento de pessoas totalmente viciadas em informações: elas precisam estar permanentemente on-line. Para esses usuários móveis, o par trançado, o cabo coaxial e a fibra óptica não têm a menor utilidade. Eles precisam transferir dados para seus laptops, notebooks, palmtops e computadores de bolso ou de pulso sem depender da infraestrutura de comunicação terrestre. A resposta para esses usuários está na comunicação sem fios.

Nas próximas seções, examinaremos os conceitos básicos da comunicação sem fios em geral, pois ela tem muitas outras aplicações importantes além de oferecer conectividade aos usuários que desejam navegar na Web enquanto estão na praia. Existem algumas outras circunstâncias em que a comunicação sem fios apresenta vantagens até mesmo para dispositivos fixos. Por exemplo, quando há dificuldades para instalar cabos de fibra óptica em um prédio, por causa de acidentes geográficos (montanhas, florestas, pântanos etc.), a tecnologia de transmissão sem fios é melhor. Não é à toa que a moderna comunicação digital sem fios teve início nas ilhas havaianas, onde os usuários estavam separados de seu centro de computação por grandes distâncias no Oceano Pacífico e onde o sistema de telefonia era totalmente inadequado.

2.3.1 O espectro eletromagnético

Quando se movem, os elétrons criam ondas eletromagnéticas que podem se propagar pelo espaço livre (até mesmo no vácuo). Essas ondas foram previstas pelo físico inglês James Clerk Maxwell em 1865 e foram observadas pela primeira vez pelo físico alemão Heinrich Hertz em 1887. O número de oscilações por segundo de uma onda eletromagnética é chamado **frequência**, f, e é medido em **Hz** (em homenagem a Heinrich Hertz). A distância entre dois pontos máximos (ou mínimos) consecutivos é chamada **comprimento de onda**, designada universalmente pela letra grega λ (lambda).

Quando se instala uma antena de tamanho apropriado em um circuito elétrico, as ondas eletromagnéticas podem ser transmitidas e recebidas com eficiência por um receptor localizado a uma distância bastante razoável. Toda a comunicação sem fios é baseada nesse princípio.

No vácuo, todas as ondas eletromagnéticas viajam à mesma velocidade, independentemente de sua frequência. Essa velocidade, geralmente chamada **velocidade da luz**, c, é aproximadamente igual a 3×10^8 m/s, ou cerca de 30 cm por nanossegundo. No cobre ou na fibra, a velocidade cai para cerca de 2/3 desse valor e se torna ligeiramente dependente da frequência. A velocidade da luz é o limite máximo que se pode alcançar. Nenhum objeto ou sinal pode se mover com maior rapidez do que ela.

A relação fundamental entre f, λ e c (no vácuo) é:

$$\lambda f = c \qquad (2.4)$$

Como c é uma constante, se conhecermos f, chegaremos a λ e vice-versa. Como uma regra prática, quando l é medido em metros e f em MHz, $\lambda f \approx 300$. Por exemplo, ondas de 100 MHz têm cerca de 3 m de comprimento, ondas de 1.000 MHz têm 0,3 metros e ondas com 0,1 metro têm uma frequência igual a 3.000 MHz.

O espectro eletromagnético é mostrado na Figura 2.8. As faixas de rádio, micro-ondas, infravermelho e luz visível do espectro podem ser usadas na transmissão de informações, por meio de modulação da amplitude, da frequência ou da fase das ondas. A luz ultravioleta, os raios X e os raios gama representariam opções ainda melhores, por terem frequências mais altas, mas são difíceis de produzir e modular, além de não se propagarem bem através dos prédios e de serem perigosos para os seres vivos. As bandas (ou faixas) de frequências listadas na parte inferior da Figura 2.8 são os nomes oficiais definidos pela ITU (International Telecommunication Union) e se baseiam nos comprimentos de onda; portanto, a banda LF vai de 1 a 10 km (aproximadamente, de 30 kHz a 300 kHz). Os termos LF, MF e HF são as abreviaturas, em inglês, de baixa, média e alta frequência, respectivamente. É claro que, quando esses nomes foram criados, ninguém esperava ultrapassar 10 MHz, de forma que foram atribuídos os seguintes nomes às bandas mais altas surgidas posteriormente: Very, Ultra, Super, Extremely e Tremendously High Frequency. Além desses não há outros nomes, mas Incredibly, Astonishingly e Prodigiously High Frequency (IHF, AHF e PHF) também serviriam muito bem.

Sabemos por Shannon (Equação 2.3) que o volume de informações que uma onda eletromagnética é capaz de transportar depende da potência recebida e está diretamente relacionado à sua largura de banda. Observando a Figura 2.8, é possível entender com clareza por que as pessoas ligadas a redes têm um carinho todo especial pelas fibras ópticas. Muitos GHz de largura de banda estão disponíveis para a transmissão de dados na banda de micro-ondas, e ainda mais na fibra, pois está mais à direita em nossa escala logarítmica. Como exemplo, considere a banda de 1,30 mícron da Figura 2.6, que tem uma largura de 0,17 micra. Se usarmos a Equação 2.4 para encontrar as frequências dos comprimentos de onda inicial e final, descobrimos que a faixa de frequência é de aproximadamente 30.000 GHz. Com uma razoável relação sinal/ruído de 10 dB, teremos 300 Tbps.

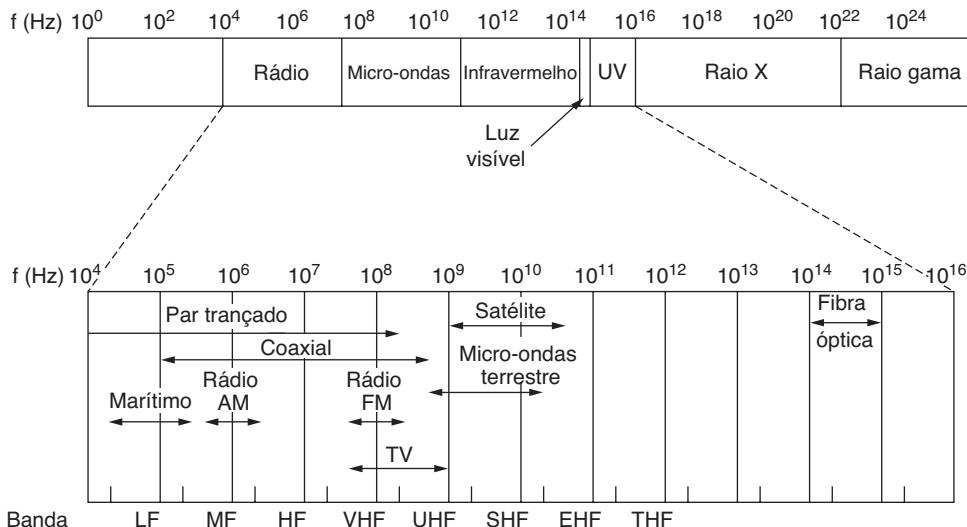



Figura 2.8 | O espectro eletromagnético e a maneira como ele é usado na comunicação.

A maioria das transmissões utiliza uma banda de frequência estreita (ou seja, $\Delta f / f \ll 1$). Elas concentram seus sinais nessa banda estreita para usar o espectro com mais eficiência e obter taxas de dados razoáveis transmitindo com potência suficiente. No entanto, em alguns casos, é usada uma banda larga, com três variações. No **espectro por salto de frequência**, o transmissor salta de uma frequência para outra centenas de vezes por segundo. Essa técnica é muito usada em comunicações militares, pois dificulta a detecção das transmissões e é praticamente impossível obstruí-las. Ela também oferece boa resistência ao enfraquecimento por múltiplos caminhos (*multipath fading*), porque o receptor não ficará restrito a uma frequência quando impossibilitada por tempo suficiente para encerrar a comunicação. Essa robustez a torna útil para as partes mais sobrecarregadas do espectro, como as bandas ISM que descreveremos em breve. Essa técnica também é aplicada comercialmente — por exemplo, no Bluetooth e nas versões mais antigas das redes 802.11.

Como curiosidade, vale a pena mencionar que uma das pessoas que criaram essa técnica foi a atriz de cinema Hedy Lamarr, a deusa austríaca do sexo, primeira mulher a aparecer nua em um filme cinematográfico (o filme tcheco *Êxtase*, de 1933). Seu primeiro marido era fabricante de armamentos e mostrou a ela como era fácil bloquear os sinais de rádio então empregados para controlar torpedos. Quando descobriu que ele estava vendendo armas a Hitler, ela ficou horrorizada, se disfarçou de criada para escapar dele e fugiu para Hollywood, a fim de continuar sua carreira como atriz de cinema. Em seu tempo livre, Hedy inventou o salto de frequência para ajudar no esforço de guerra dos Aliados. Seu esquema utilizava 88 frequências, o número de teclas (e frequências) do piano. Por sua invenção, ela e seu amigo, o compositor George Antheil, receberam a patente 2.292.387 dos Estados Unidos. Porém, eles não conseguiram convencer a Marinha americana de que sua invenção tinha alguma utilidade prática, e nunca receberam royalties por ela. Somente anos depois de expirar a patente, a invenção se tornou popular.

A segunda forma de espectro de dispersão, o **espectro de dispersão de sequência direta**, usa uma sequência de código para dispersar o sinal de dados por uma banda de frequência mais ampla. Ela é bastante usada comercialmente, como um meio eficiente na utilização do espectro, para permitir que vários sinais compartilhem a mesma banda de frequência. Esses sinais podem receber diferentes códigos, um método chamado **CDMA** (**Code Division Multiple Access**), ao qual retornaremos mais adiante neste capítulo. Tal método aparece em contraste com o salto de frequência na Figura 2.9. Ele forma a base das redes de telefonia móvel 3G e também é usado no GPS (*Global Positioning System*). Mesmo sem códigos diferentes, o espectro de dispersão de sequência direta, assim como o espectro de dispersão de salto de frequência, pode tolerar interferência de banda estreita e enfraquecimento por múltiplos caminhos, pois apenas uma fração do sinal desejado é perdida. Ele é usado nesse papel nas LANs sem fio 802.11b mais antigas. Para obter informações mais detalhadas e fascinantes sobre a história da comunicação por espectro de dispersão, consulte Scholtz (1982).

Um terceiro método de comunicação com uma banda mais larga é a comunicação **UWB** (**Ultra-WideBand**). A UWB envia uma série de pulsos rápidos, variando suas posições para trocar informações. As rápidas transições levam a um sinal que se espalha estreitamente por uma banda de frequência muito larga. Esse método é definido como sinais que têm uma largura de banda de pelo menos 500 MHz ou pelo menos 20 por cento da frequência central de sua

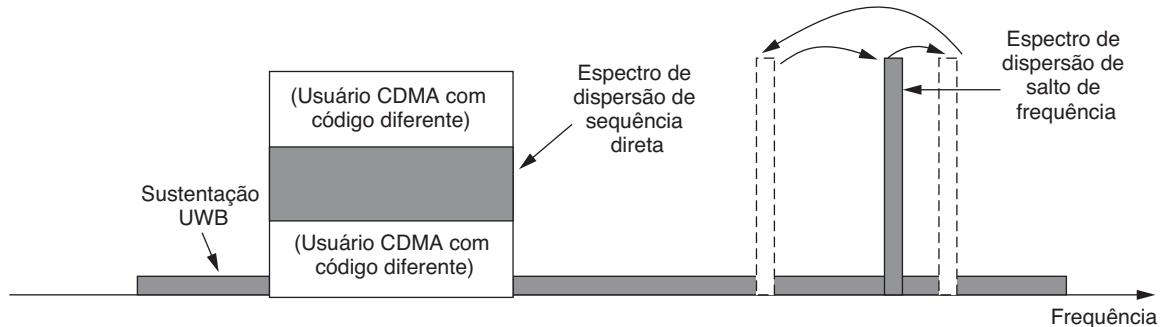



Figura 2.9 | Comunicação por espectro de dispersão e ultra-wideband (UWB).

banda de frequência. A UWB também aparece na Figura 2.9. Com tanta largura de banda, a UWB tem o potencial de se comunicar em taxas altas. Por se espalhar por uma ampla banda de frequências, ela pode tolerar uma grande quantidade de interferência relativamente forte de outros sinais de banda estreita. Tão importante quanto isso, como a UWB tem pouquíssima energia em qualquer frequência quando usada para transmissão em curta distância, ela não causa interferência prejudicial para outros sinais de rádio de banda estreita. Dizemos que ela **sustenta** os outros sinais. Essa coexistência pacífica proporcionou sua aplicação em PANs sem fio que trabalham em até 1 Gbps, embora o sucesso comercial tenha sido misto. Ela também pode ser usada para formar imagens através de objetos sólidos (solo, paredes e corpos), ou como parte de sistemas de localização precisos.

A seguir, mostraremos como as diversas partes do espectro eletromagnético da Figura 2.9 são usadas, começando pelo rádio. Partiremos da premissa de que todas as transmissões utilizam uma banda de frequência estreita, a menos que seja dito de outra forma.

2.3.2 Transmissão de rádio

As ondas de rádio são fáceis de gerar, podem percorrer longas distâncias e penetrar facilmente nos prédios; portanto, são amplamente utilizadas para comunicação, seja em ambientes fechados, seja em locais abertos. As ondas de rádio também são omnidirecionais, o que significa que elas viajam em todas as direções a partir da origem; desse modo, o transmissor e o receptor não precisam estar cuidadosa e fisicamente alinhados.

Vale lembrar que o rádio omnidirecional nem sempre é bom. Na década de 1970, a General Motors decidiu equipar todos os seus novos Cadillacs com freios controlados por computador, que impediam o travamento das rodas. Quando o motorista pisava no pedal de freio, o computador prendia e soltava os freios, em vez de travá-los de verdade. Um belo dia, um guarda rodoviário de Ohio começou a usar seu novo rádio móvel para falar com a central de polícia e, de repente, o Cadillac próximo a ele passou a se comportar como um cavalo selvagem. Depois de ser abordado pelo patrulheiro, o motorista disse que não tinha feito nada e que o carro tinha ficado louco de uma hora para outra.

Com o tempo, começou a surgir um padrão: às vezes, os Cadillacs enlouqueciam, mas somente quando trafegavam pelas principais estradas de Ohio, particularmente quando estavam sendo observados por um guarda rodoviário. A General Motors demorou a entender o motivo pelo qual os Cadillacs funcionavam sem problemas nos outros Estados e também em rodovias secundárias de Ohio. Só depois de muita pesquisa, eles descobriram que a fiação do Cadillac formava uma ótima antena que captava a frequência usada pelo novo sistema de rádio da Patrulha Rodoviária de Ohio.

As propriedades das ondas de rádio dependem da frequência. Em baixas frequências, as ondas de rádio atravessam bem os obstáculos, mas a potência cai abruptamente à medida que a distância da origem aumenta — pelo menos cerca de $1/r^2$ no ar —, pois a energia do sinal se espalha de forma mais estreita por uma superfície maior. Essa atenuação é chamada **perda no caminho**. Em altas frequências, as ondas de rádio tendem a viajar em linha reta e a ricochetear nos obstáculos. A perda do caminho ainda reduz a potência, embora o sinal recebido também possa depender muito das reflexões. Ondas de rádio de alta frequência também são absorvidas pela chuva e outros obstáculos, até certo ponto, mais do que as frequências baixas. Em todas as frequências, as ondas de rádio estão sujeitas à interferência de motores e outros equipamentos elétricos.

É interessante comparar a atenuação das ondas de rádio com a dos sinais nos meios guiados. Com fibra, cabo coaxial e par trançado, o sinal cai pela mesma fração por distância unitária, por exemplo, 20 dB por 100 m para o par trançado. Com o rádio, o sinal cai pela mesma fração enquanto a distância dobra, por exemplo, 6 dB por duplicação no espaço livre. Isso significa que as ondas de rádio podem percorrer longas distâncias, e a interferência entre os usuários é um problema. Por essa razão, todos os governos exercem um rígido controle sobre o licenciamento do uso de transmissores de rádio, com apenas uma exceção, descrita mais adiante neste capítulo.

Nas bandas VLF, LF e MF, as ondas de rádio se propagam perto do solo, como mostra a Figura 2.10(a). Essas ondas podem ser detectadas dentro de um raio de, talvez, mil quilômetros nas frequências mais baixas; nas frequências mais altas, esse raio é menor. A radiodifusão em frequências AM utiliza a banda MF, razão pela qual as ondas de rádio produzidas pelas estações de rádio AM de Boston não podem ser captadas facilmente em Nova York. As ondas de rádio nessas bandas atravessam os prédios com facilidade; esse é o motivo por que os rádios portáteis funcionam em ambientes fechados. O principal problema relacionado à utilização dessas bandas para comunicação de dados diz respeito à baixa largura de banda que oferecem (ver Equação 2.4).

Nas bandas HF e VHF, as ondas que se propagam ao longo do solo tendem a ser absorvidas pela terra. No entanto, as ondas que alcançam a ionosfera, uma camada de partículas carregadas situadas em torno da Terra a uma altura de 100 a 500 km, são refratadas por ela e enviadas de volta à Terra, como mostra a Figura 2.10(b). Em determinadas condições atmosféricas, os sinais podem ricochetear diversas vezes. Os operadores de radioamador utilizam essas bandas em comunicações de longa distância. Os militares também se comunicam nas bandas HF e VHF.

2.3.3 Transmissão de micro-ondas

Acima de 100 MHz, as ondas trafegam praticamente em linha reta e, portanto, podem ser concentradas em uma faixa estreita. A concentração de toda a energia em um pequeno feixe através de uma antena parabólica (como a conhecida antena de TV por satélite) oferece uma relação sinal/ruído muito mais alta, mas as antenas de transmissão e recepção devem estar alinhadas com o máximo de precisão. Além disso, essa direcionalidade permite o alinhamento de vários transmissores em uma única fileira, fazendo com que se comuniquem com vários receptores também alinhados sem que haja interferência, desde que sejam observadas algumas regras mínimas de espaçamento. Antes da fibra óptica, durante décadas essas micro-ondas formaram o núcleo do sistema de transmissão telefônica de longa distância. Na verdade, a MCI, uma das primeiras concorrentes da AT&T após sua desregulamentação, construiu todo o seu sistema com comunicações de micro-ondas que percorriam dezenas de quilômetros entre uma torre e outra. Até mesmo o nome da empresa refletia isso (MCI significava Microwave Communications, Inc.). Há muito tempo, a MCI passou a utilizar as fibras ópticas e, por meio de uma longa série de fusões corporativas e falências no setor de telecomunicações, passou a fazer parte da Verizon.

Tendo em vista que as micro-ondas viajam em linha reta, se as torres estiverem muito afastadas, a Terra acabará ficando entre elas (como acontece no caso de um enlace entre São Francisco e Amsterdã). Assim, é preciso instalar repetidores em intervalos periódicos. Quanto mais altas as torres, mais distantes elas podem estar umas das outras. A distância entre os repetidores aumenta de acordo com a raiz quadrada da altura da torre. Torres com 100 m de altura devem ter repetidores a cada 80 km.

Ao contrário das ondas de rádio nas frequências mais baixas, as micro-ondas não atravessam muito bem as paredes dos prédios. Além disso, muito embora o feixe possa estar bem concentrado no transmissor, ainda há alguma divergência no espaço. Algumas ondas podem ser refratadas nas camadas atmosféricas mais baixas e, consequentemente, sua chegada pode ser mais demorada que a das ondas diretas. As ondas atrasadas podem chegar fora de fase em relação à onda direta, e assim cancelar o sinal. Esse efeito é chamado **enfraquecimento por múltiplos caminhos** (**multipath fading**) e em geral é um problema sério. Ele depende das condições atmosféricas e da frequência. Algumas operadoras mantêm 10 por cento de seus canais ociosos como sobressalentes, para quando o enfraquecimento por múltiplos caminhos eliminar temporariamente alguma banda de frequência.

A demanda por mais e mais espectro incentiva as operadoras a usarem transmissões com frequências cada vez mais altas. As bandas de até 10 GHz agora são de uso rotineiro, mas a partir de 4 GHz surge um novo problema: a absorção pela água. Essas ondas têm apenas alguns centímetros de comprimento e são absorvidas pela chuva. Esse efeito não causaria nenhum problema se estivéssemos planejando construir um gigantesco forno de micro-ondas para ser usado a céu aberto mas, no caso das comunicações, trata-se de um grave problema. Assim como acontece

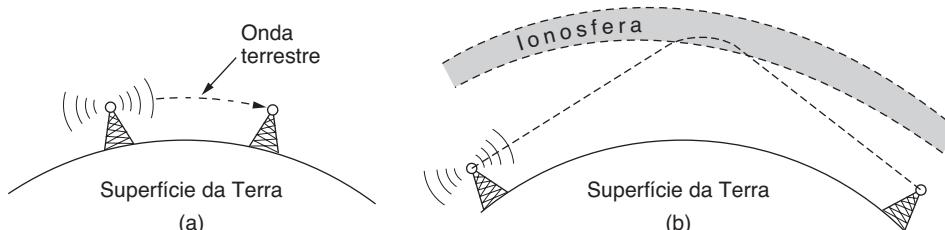



Figura 2.10 | (a) Nas bandas VLF, VF e MF, as ondas de rádio obedecem à curvatura da Terra. (b) Na banda HF, elas ricocheteiam na ionosfera.

com o enfraquecimento por múltiplos caminhos, a única solução é desligar os enlaces que estão sendo afetados pela chuva e criar uma nova rota que os contorne.

Em resumo, a comunicação por micro-ondas é muito usada na telefonia de longa distância, em telefones celulares, na distribuição de sinais de televisão e em outros usos que uma severa diminuição do espectro obrigou a desenvolver. Ela tem uma série de vantagens significativas sobre a fibra. A mais importante delas é que as micro-ondas dispensam a necessidade de se ter direitos sobre um caminho. Além do mais, quando se compra um pequeno lote de terra a cada 50 quilômetros e nele é instalada uma torre de micro-ondas, é possível ignorar o sistema telefônico e se comunicar diretamente. Foi por essa razão que a MCI passou a trabalhar tão rapidamente como uma companhia telefônica de longa distância. (A Sprint, outro antigo concorrente da AT&T, trilhou um caminho bem diferente: ela se formou a partir da Southern Pacific Railroad, que já detinha um grande número de concessões de direitos de percurso, e simplesmente enterrava os cabos de fibra ao lado das ferrovias.)

O uso de micro-ondas também é relativamente econômico. A instalação de duas torres simples (com alguns postes com quatro esteios) e a colocação de antenas em cada uma delas pode ser menos dispendiosa que enterrar 50 quilômetros de fibra em uma área urbana congestionada ou em uma região montanhosa, e talvez seja mais econômica que arrendar a rede de fibra da companhia telefônica, especialmente se esta ainda não tiver coberto totalmente os custos da retirada do cobre quando os cabos de fibra foram instalados.

A política do espectro eletromagnético

Para evitar o caos total, têm sido feitos acordos nacionais e internacionais a respeito de quem terá o direito de usar cada uma das frequências. Como todos querem uma taxa de dados mais alta, todos desejam um espectro maior. Os governos nacionais alocam bandas do espectro para rádios AM e FM, televisão e telefones celulares, assim como para as empresas de telefonia, a polícia, os usuários marítimos, de navegação, militares, do governo e para muitos outros usuários concorrentes. Em termos mundiais, uma agência da ITU-R (WARC) tenta coordenar essa alocação de forma que possam ser fabricados dispositivos que funcionem em vários países. Porém, os países não são limitados pelas recomendações da ITU-R, e a FCC (Federal Communications Commission), que realiza a alocação para os Estados Unidos, ocasionalmente tem rejeitado essas recomendações (em geral porque elas exigiam que algum grupo politicamente poderoso desistisse de alguma fração do espectro).

Até mesmo quando uma parte do espectro é alocada para algum uso, como telefones celulares, existe a questão adicional de decidir qual concessionária terá permissão para usar quais frequências. Três algoritmos foram extensamente usados no passado. O algoritmo mais antigo, frequentemente chamado **concurso de beleza**, exige que cada concessionária explique por que sua proposta serve melhor ao interesse público. Então, os funcionários do governo decidem qual dentre as belas histórias mais lhes agrada. Fazer algum funcionário do governo oferecer como prêmio a propriedade de bilhões de dólares à sua empresa favorita em geral leva a suborno, corrupção, nepotismo e a crimes piores. Além disso, até mesmo um funcionário do governo honesto e escrupuloso que imagine que uma companhia estrangeira poderia realizar um trabalho melhor que qualquer das empresas nacionais teria muito a explicar.

Essa observação levou ao algoritmo 2, que realiza um **sorteio** entre as empresas interessadas. O problema com essa ideia é que empresas que não têm nenhum interesse em usar o espectro podem participar desse sorteio. Se, digamos, um restaurante ou uma cadeia de sapatarias ganhasse, a empresa poderia revender o espectro a uma concessionária com um enorme lucro e sem nenhum risco.

A ideia de conceder fatias do espectro a empresas com uma enorme dose de sorte mas sem nenhum método tem sido severamente criticada por muitos, o que levou ao algoritmo 3: realizar **leilões** e conceder a largura de banda à empresa que fizer a melhor proposta. Quando a Inglaterra leiloou as frequências necessárias para os sistemas de telefonia móvel em 2000, o governo esperava obter aproximadamente 4 bilhões de dólares. Na realidade, recebeu cerca de 40 bilhões de dólares, pois as concessionárias entraram em uma disputa frenética, mortas de medo de perder o barco da telefonia móvel. Esse evento despertou a ganância dos governos vizinhos e os inspirou a realizar seus próprios leilões. Isso funcionou, mas também deixou algumas concessionárias tão endividadas que elas chegaram perto da falência. Até mesmo nos melhores casos, muitos anos serão necessários para essas empresas recuperarem o custo do licenciamento.

Uma abordagem muito diferente para alocar frequências é simplesmente não alocá-las. Em vez disso, basta deixar todo mundo transmitir à vontade, mas regular a potência utilizada, de forma que as estações tenham um alcance tão pequeno que não possam interferir umas com as outras. De acordo com essa proposta, a maioria dos governos reserva algumas bandas de frequência, chamadas bandas **ISM** (**Industrial, Scientific, Medical**) para uso sem licença. Sistemas para abertura de portas de garagens, telefones sem fio, brinquedos controlados por rádio, dispositivos tipo mouse sem fio e vários outros aparelhos domésticos sem fios utilizam as bandas ISM. Para minimizar a interferência entre esses dispositivos não coordenados, a FCC estabelece que todos os dispositivos nas bandas ISM devem limitar sua potência de transmissão (por exemplo, para 1 watt) e usar outras técnicas para dispersar seus sinais por uma faixa de frequências. Os dispositivos também podem ter de evitar interferência com instalações de radar.

A localização das bandas ISM varia um pouco de país para país. Por exemplo, nos Estados Unidos, dispositivos cuja potência está abaixo de 1 watt podem usar as bandas mostradas na Figura 2.11 sem exigir uma licença da FCC. A banda de 900 MHz era usada para as primeiras versões do 802.11, mas está sobrecarregada. A banda de 2,4 GHz está disponível na maioria dos países e é bastante usada para 802.11b/g e Bluetooth, mas é sujeita a interferências de fornos de micro-ondas e instalações de radar. A banda de 5 GHz do espectro inclui bandas **U-NII** (**Unlicensed National Information Infrastructure**). As bandas de 5 GHz são relativamente pouco desenvolvidas, mas, por terem a maior largura de banda e serem usadas pelo 802.11a, rapidamente estão se tornando mais populares.

As bandas não licenciadas foram um grande sucesso na década passada. A capacidade de usar o espectro livremente ocasionou uma grande inovação nas LANs e PANs sem fios, evidenciada pela implantação generalizada de tecnologias como 802.11 e Bluetooth. Para continuar essa inovação, é necessário mais espectro. Um desenvolvimento interessante nos Estados Unidos é a decisão da FCC, em 2009, de permitir o uso não licenciado de **espaços vazios** em torno de 700 MHz. Os espaços vazios são bandas de frequência que foram alocadas mas não estão sendo usadas localmente. A mudança das transmissões de televisão de analógicas para digitais nos Estados Unidos, em 2010, liberaram os espaços vazios em torno de 700 MHz. A única dificuldade é que, para usar os espaços vazios, dispositivos não licenciados precisam poder detectar quaisquer transmissores licenciados vizinhos, incluindo microfones sem fio, que têm os direitos iniciais de usar a banda de frequência.

Outra atividade intensa está acontecendo em torno da banda de 60 GHz. A FCC abriu de 57 GHz a 64 GHz para operação não licenciada em 2001. Essa faixa é uma parte enorme do espectro, mais do que todas as outras bandas ISM combinadas, de modo que pode aceitar o tipo de rede de alta velocidade que seria necessária para enviar fluxo de TV de alta definição através do ar a sua sala de estar. A 60 GHz, as ondas de rádio são absorvidas pelo oxigênio. Isso significa que os sinais não se propagam longe, tornando-os bem adequados a redes de curta distância. As altas frequências (60 GHz é uma banda de frequência extremamente alta, ou de 'milímetro', logo abaixo da radiação de infravermelho) representaram um desafio inicial para os fabricantes de equipamentos, mas os produtos atualmente estão no mercado.

2.3.4 Transmissão em infravermelho

As ondas de infravermelho não guiadas são extensamente utilizadas na comunicação de curto alcance. Todos os dispositivos de controle remoto utilizados nos aparelhos de televisão, videocassetes e equipamentos estereofônicos empregam a comunicação por infravermelho. Eles são relativamente direcionais, econômicos e fáceis de montar, mas têm uma desvantagem importante: não atravessam objetos sólidos (para provar essa afirmação, posicione-se entre o controle remoto e o televisor e veja se ele funciona). Em geral, quando nos deslocamos do rádio de onda longa em direção à luz visível, as ondas assumem um comportamento cada vez mais parecido com o da luz, perdendo pouco a pouco as características de ondas de rádio.

Por outro lado, o fato de as ondas de infravermelho não atravessarem paredes sólidas pode ser visto como uma qualidade. É por essa razão que um sistema infravermelho instalado em um ambiente fechado não interfere em um sistema semelhante instalado nas salas ou nos prédios adjacentes: não é possível controlar o aparelho de televisão do vizinho com o seu controle remoto. Além disso, a segurança dos sistemas de infravermelho contra bisbilhotagem é melhor que a dos sistemas de rádio, exatamente por essa razão. Portanto, não é necessária nenhuma licença do governo para operar um sistema de infravermelho, ao contrário dos sistemas de rádio, que devem ser licenciados

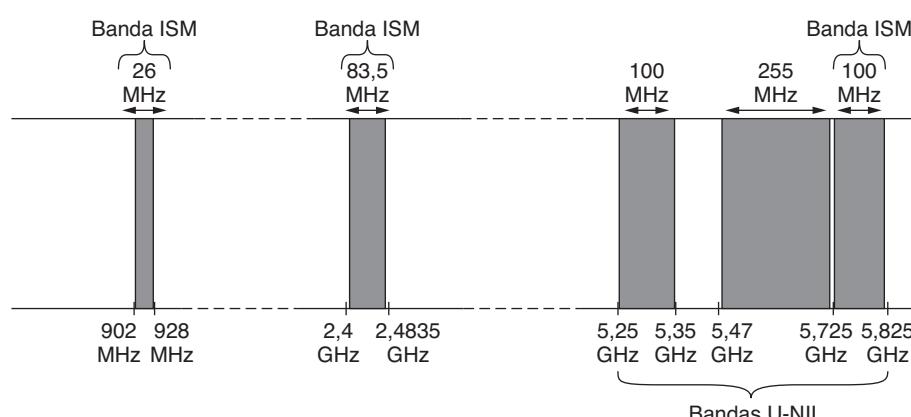



Figura 2.11 | As bandas ISM e U-NII usadas nos Estados Unidos pelos dispositivos sem fio.

fora das bandas ISM. A comunicação por infravermelho tem uso limitado em escritórios, por exemplo, para conectar notebooks e impressoras com o padrão **IrDA** (**Infrared Data Association**), mas não deverá ter um papel importante no jogo das comunicações.

2.3.5 Transmissão via luz

A transmissão óptica não guiada, ou **óptica do espaço livre**, vem sendo utilizada há séculos. Uma aplicação mais moderna consiste em conectar as LANs em dois prédios por meio de lasers instalados em seus telhados. Por sua própria natureza, a transmissão óptica usando raios laser é unidirecional; assim, cada prédio precisa de seu próprio raio laser e de seu próprio fotodetector. Esse esquema oferece uma largura de banda muito alta e é relativamente seguro, pois é difícil interceptar um raio laser estreito. Ele também é relativamente fácil de ser instalado e, ao contrário das micro-ondas, não precisa de uma licença da FCC.

A potência do laser, um feixe muito estreito, também é uma desvantagem aqui. Apontar um raio laser de 1 mm para um alvo do tamanho de uma cabeça de alfinete a 500 metros de distância exige a mira de um herói de faroeste. Normalmente, são colocadas lentes no sistema para tirar um pouco do foco do raio. Para aumentar a dificuldade, mudanças no vento e na temperatura podem distorcer o raio, e os feixes de raios laser de fato não podem atravessar chuva ou neblina espessa, mas normalmente funcionam bem em dias ensolarados. Contudo, muitos desses fatores não são problemas quando o uso é para conectar duas naves espaciais.

Um dos autores deste livro (AST) certa vez participou de uma conferência em um moderno hotel europeu cujos organizadores tiveram a felicidade de oferecer uma sala repleta de terminais para que os participantes pudessem ler suas mensagens de correio eletrônico durante as apresentações menos interessantes. Como o PTT local não se dispôs a instalar um grande número de linhas telefônicas, que após três dias seriam desativadas, os organizadores colocaram um raio laser no telhado e o apontaram na direção do prédio de ciência da computação da universidade onde trabalhavam, situada a alguns quilômetros dali. Eles testaram o sistema na noite anterior à conferência e ele havia funcionado perfeitamente. Às 9h da manhã seguinte, em um belo dia de sol, o sistema entrou em pane e ficou fora do ar durante todo o dia. Nos dois dias seguintes, o problema se repetiu. Após a conferência, os organizadores descobriram o problema. O calor do sol fez com que emanassem correntes de convecção do telhado do prédio, como mostra a Figura 2.12. Esse ar turbulento desviou o feixe e fez com que ele dançasse em torno do detector, algo parecido com uma estrada tremulante em um dia muito quente. A lição aqui é que, para funcionar bem em condições difíceis e também em boas condições, os enlaces ópticos não guiados precisam ser elaborados com uma margem de erro suficiente.

Atualmente, a comunicação óptica não guiada pode parecer uma tecnologia de rede exótica, mas logo pode-

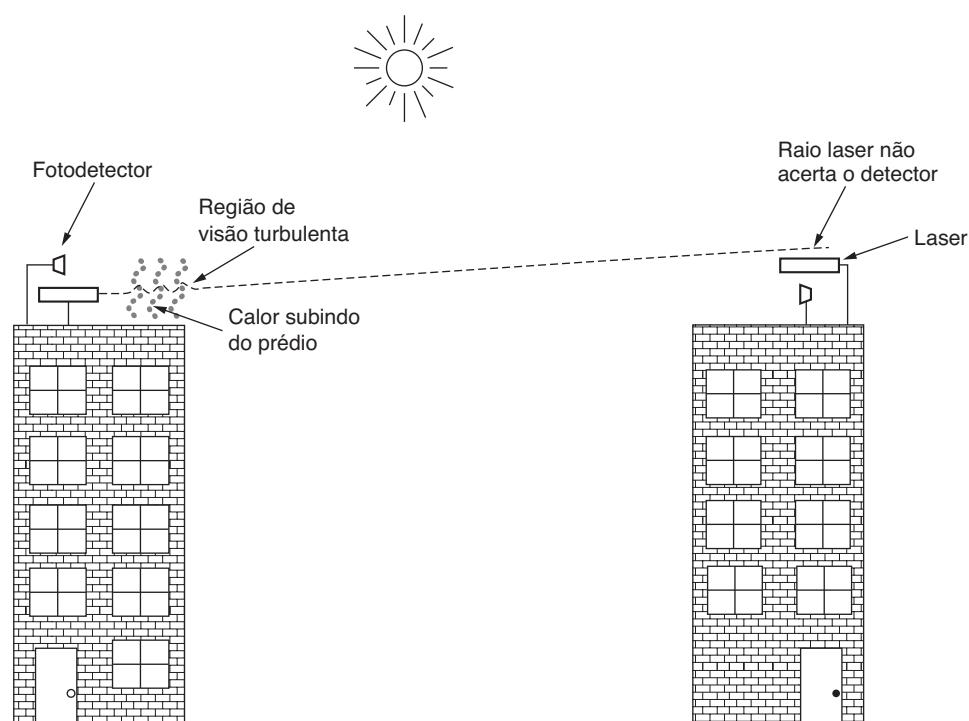



Figura 2.12 | Correntes de convecção podem interferir nos sistemas de comunicação a laser. A figura mostra um sistema bidirecional com dois lasers.

rá se tornar mais predominante. Estamos cercados de câmeras (que sentem a luz) e telas (que emitem luz usando LEDs ou outra tecnologia). A comunicação de dados pode ser disposta em cima dessas telas codificando informações no padrão em que os LEDs se acendem e apagam, abaixo do limiar da percepção humana. A comunicação via luz visível dessa maneira é inerentemente segura e cria uma rede de baixa velocidade na vizinhança imediata da tela. Isso poderia permitir todo o tipo de cenário de computação ubíqua. As luzes piscando nos veículos de emergência podem alertar os sinais de trânsito e veículos mais próximos para ajudar a limpar um caminho. Os sinais de informação poderiam transmitir mapas por difusão. As lâmpadas em ocasiões festivas poderiam até mesmo transmitir canções sincronizadas com sua exibição.

2.4 Satélites de comunicações

Na década de 1950 e no início dos anos 1960, as pessoas tentavam configurar sistemas de comunicações emitindo sinais que se refletiam em balões meteorológicos metalizados. Infelizmente, os sinais recebidos eram muito fracos para que tivessem algum uso prático. Em seguida, a Marinha dos Estados Unidos detectou uma espécie de balão meteorológico que ficava permanentemente no céu — a Lua — e criou um sistema operacional para comunicações entre o navio e a base, utilizando a Lua em suas transmissões.

O progresso no campo da comunicação celeste precisou esperar até que o primeiro satélite de comunicações fosse lançado. A principal diferença entre um satélite artificial e um real é que o artificial amplifica os sinais antes de enviá-los de volta, transformando uma estranha curiosidade em um avançado sistema de comunicações.

Os satélites de comunicações possuem algumas propriedades interessantes, que os tornam atraentes para muitas aplicações. Em sua forma mais simples, um satélite de comunicações pode ser considerado um grande repetidor de micro-ondas no céu. Ele contém diversos **transponders**; cada um deles ouve uma parte do espectro, amplifica os sinais de entrada e os transmite novamente em outra frequência, para evitar interferência com o sinal de entrada. Esse modo de operação é conhecido como um **canal em curva** (**bent pipe**). O processamento digital pode ser acrescentado para manipular ou redirecionar separadamente os feixes de dados na banda geral, ou informações digitais ainda podem ser recebidas pelo satélite e retransmitidas. A regeneração de sinais dessa maneira melhora o desempenho em comparação com um canal em curva, pois o satélite não amplifica o ruído no sinal ascendente. Os feixes descendentes podem ser largos, cobrindo uma fração substancial da superfície terrestre, ou estreitos, cobrindo uma área com apenas centenas de quilômetros de diâmetro.

De acordo com a lei de Kepler, o período orbital de um satélite varia de acordo com o raio da órbita elevado à potência 3/2. Quanto mais alto o satélite, mais longo o período. Perto da superfície da Terra, o período é de cerca de 90 minutos. Consequentemente, os satélites de órbita baixa saem de visão com bastante rapidez; assim, são necessários muitos deles para proporcionar cobertura contínua, e as antenas terrestres precisam acompanhá-los. A uma altitude de aproximadamente 35.800 km, o período é de 24 horas. Em uma altitude de 384.000 km, o período é de cerca de um mês, como pode atestar qualquer pessoa que observe a Lua regularmente.

O período do satélite é importante, mas não é o único fator para determinar onde posicioná-lo. Outra questão é a presença dos cinturões de Van Allen, camadas de partí-

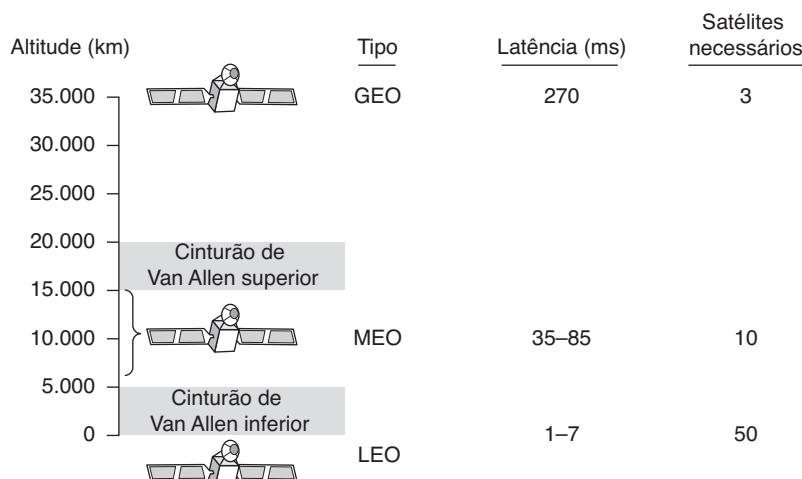



Figura 2.13 | Satélites de comunicações e algumas de suas propriedades, inclusive altitude acima da Terra, tempo de atraso de ida e volta, e o número de satélites necessários para cobertura global.

culas altamente carregadas que são capturadas pelo campo magnético terrestre. Qualquer satélite em órbita dentro deles seria destruído com bastante rapidez pelas partículas carregadas com alta energia. Esses fatores nos levam a identificar três regiões nas quais os satélites podem ser posicionados com segurança. Essas regiões e algumas de suas propriedades estão ilustradas na Figura 2.13. A seguir, descreveremos rapidamente os satélites que habitam cada uma dessas regiões.

2.4.1 Satélites geoestacionários

Em 1945, o escritor de ficção científica Arthur C. Clarke calculou que um satélite na altitude de 35.800 km em uma órbita circular equatorial pareceria permanecer imóvel no céu, e assim não precisaria ser rastreado (Clarke, 1945). Ele continuou a descrever um sistema de comunicação completa que usava esses **satélites geoestacionários** (tripulados), incluindo as órbitas, os painéis solares, as frequências de rádio e os procedimentos de lançamento. Infelizmente, ele concluiu que os satélites eram impraticáveis em virtude da impossibilidade de colocar em órbita amplificadores a válvulas, frágeis e gastadores de energia; assim, nunca levou sua ideia adiante, embora tenha escrito algumas histórias de ficção científica sobre ela.

A invenção do transistor mudou tudo isso, e o primeiro satélite artificial de comunicações, chamado Telstar, foi lançado em julho de 1962. Desde então, os satélites de comunicações se transformaram em um negócio de vários bilhões de dólares, e o único aspecto do espaço sideral que se tornou altamente lucrativo. Esses satélites de alta órbita normalmente são chamados **satélites geoestacionários**, ou **GEO** (**Geoestationary Earth Orbit**).

Com a tecnologia atual, não é muito inteligente ter satélites geoestacionários com espaçamento muito menor que 2 graus entre eles no plano equatorial de 360 graus, a fim de evitar interferência. Com um espaçamento de 2 graus, só pode haver 360/2 = 180 desses satélites no céu ao mesmo tempo. No entanto, cada transponder pode usar várias frequências e polarizações, com a finalidade de aumentar a largura de banda disponível.

Para evitar o caos total no céu, a alocação de slots de órbitas é feita pela ITU. Esse processo é altamente político, com países que mal saíram da idade da pedra exigindo 'seus' slots de órbitas (com a finalidade de arrendá-los pela melhor oferta). Contudo, outros países sustentam que os direitos nacionais de propriedade não se estendem para cima até a Lua e que nenhum país tem direito legal sobre os slots de órbita acima de seu território. Para aumentar a disputa, as telecomunicações comerciais não são a única aplicação. Emissoras de televisão, governos e instituições militares também querem ter uma fatia dessa torta orbital.

Os satélites modernos podem ser bastante grandes, pesando até 5.000 kg e consumindo vários quilowatts de energia elétrica produzida pelos painéis solares. Os efeitos da gravidade solar, lunar e planetária tendem a movê-los para fora de seus slots de órbita e de suas orientações, um efeito compensado por motores de foguetes a bordo. Essa atividade de ajuste fino é chamada **manutenção da estação**. Porém, quando o combustível para os motores tiver se esgotado (em geral no período de dez anos), o satélite fica sem controle, e, portanto, tem de ser desativado. Por fim, a órbita decai, o satélite entra de novo na atmosfera e é totalmente queimado (muito raramente, ele colide com a Terra).

Os slots de órbita não são o único ponto de discórdia. As frequências também o são, porque as transmissões do satélite para a Terra (downlink) interferem com usuários de micro-ondas. Consequentemente, a ITU alocou certas bandas de frequência para usuários de satélites. As principais estão listadas na Tabela 2.3. A banda C foi a primeira a ser designada para tráfego comercial de satélite. Duas faixas de frequências são atribuídas nessa banda, a inferior para tráfego downlink (descendo do satélite) e a superior para tráfego uplink (subindo para o satélite). Para permitir que o tráfego ocorra em ambos os sentidos ao mesmo tempo, são necessários dois canais, um para cada sentido. Esses canais já estão sobrecarregados, porque também são usados pelas concessionárias de telecomunicações nos enlaces terrestres de micro-ondas. As bandas L e S foram acrescentadas por um acordo internacional em 2000. No entanto, elas são estreitas e também estão lotadas.

Banda	Downlink	Uplink	Largura de banda	Problemas
L	1,5 GHz	1,6 GHz	15 MHz	Baixa largura de banda; lotada
S	1,9 GHz	2,2 GHz	70 MHz	Baixa largura de banda; lotada
C	4,0 GHz	6,0 GHz	500 MHz	Interferência terrestre
Ku	11 GHz	14 GHz	500 MHz	Chuva
Ka	20 GHz	30 GHz	3.500 MHz	Chuva; custo do equipamento

Tabela 2.3 Principais bandas de satélite.

A próxima banda mais alta disponível para concessionárias de telecomunicações comerciais é a Ku (K under). Essa banda (ainda) não está congestionada e, nessas frequências, os satélites podem ficar à distância de apenas 1 grau. Entretanto, existe outro problema: a chuva. A água absorve bastante essas micro-ondas curtas. Felizmente, em geral as tempestades fortes costumam ser localizadas; assim, o uso de várias estações terrestres separadas por uma grande distância, em lugar de apenas uma, contorna o problema, mas ao preço de antenas, cabos e equipamentos eletrônicos extras para permitir a comutação rápida entre estações. Na banda Ka (K above), também foi alocada uma largura de banda para o tráfego de satélite comercial, mas o equipamento necessário para usá-la ainda continua sendo caro. Além dessas bandas comerciais, também existem muitas bandas governamentais e militares.

Um satélite moderno tem cerca de quarenta transponders, cada um com uma largura de banda de 36 MHz. Em geral, cada transponder opera como um canal em curva, mas satélites recentes têm alguma capacidade de processamento a bordo, permitindo uma operação mais sofisticada. Nos primeiros satélites, a divisão dos transponders em canais era estática: a largura de banda simplesmente era dividida em bandas de frequências fixas. Hoje em dia, o feixe de cada transponder é dividido em slots de tempo, com diversos usuários alternando atividades. Estudaremos essas duas técnicas (a multiplexação por divisão de frequência e a multiplexação por divisão de tempo) em detalhes mais adiante neste capítulo.

Os primeiros satélites geoestacionários tinham um único feixe espacial que iluminava cerca de 1/3 da superfície da Terra, denominado sua **área de cobertura** (**footprint**). Com o enorme declínio de preço, tamanho e requisitos de potência dos equipamentos microeletrônicos, uma estratégia de transmissão muito mais sofisticada tornou-se viável. Cada satélite é equipado com diversas antenas e vários transponders. Cada feixe descendente pode ser focalizado em uma pequena área geográfica; portanto, podem acontecer diversas transmissões ascendentes e descendentes ao mesmo tempo. Em geral, esses chamados **feixes pontuais** têm forma elíptica e podem ter apenas algumas centenas de quilômetros de diâmetro. Em geral, um satélite de comunicações para os Estados Unidos tem um único feixe para os 48 estados contíguos, além de feixes pontuais para o Alasca e o Havaí.

Um novo desenvolvimento no mundo dos satélites de comunicações é a criação de microestações de baixo custo, às vezes chamadas **VSATs** (**Very Small Aperture Terminals**) (Abramson, 2000). Esses pequenos terminais têm antenas de 1 metro ou menos (em comparação com dez metros para uma antena de GEO padrão) e podem emitir cerca de 1 watt de energia. Geralmente, o uplink é adequado para 1 Mbps, mas o downlink normalmente exige vários megabits/s. A televisão transmitida por satélite utiliza essa tecnologia na transmissão de mão única.

Em muitos sistemas VSAT, as microestações não têm energia suficiente para se comunicarem diretamente umas com as outras (via satélite, é óbvio). Em vez disso, é necessária uma estação terrestre especial, o **hub**, com uma grande antena de alto ganho para retransmitir o tráfego entre VSATs, como mostra a Figura 2.14. Nesse modo de operação, o transmissor ou o receptor possuem uma gran-

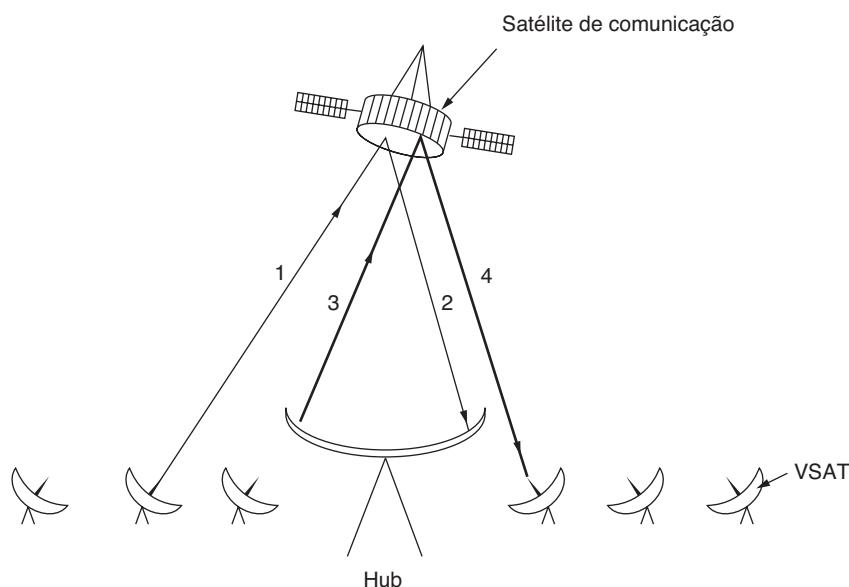



Figura 2.14 | VSATs utilizando um hub.

de antena e um amplificador de grande potência. O compromisso é um atraso mais longo em troca de estações mais econômicas para o usuário final.

Os VSATs apresentam um grande potencial em áreas rurais. Ele não é amplamente apreciado, porém mais da metade da população do mundo vive a uma distância de no máximo uma hora a pé do telefone mais próximo. Estender fios telefônicos até milhares de pequenas aldeias é algo que vai muito além do orçamento da maioria dos governos do Terceiro Mundo, mas a instalação de antenas VSAT de 1 metro de diâmetro, alimentadas por células solares, geralmente é algo viável. Os VSATs fornecem a tecnologia que irá conectar o mundo.

Os satélites de comunicações têm diversas propriedades radicalmente diferentes dos enlaces terrestres ponto a ponto. Para começar, embora os sinais enviados e recebidos por um satélite trafeguem à velocidade da luz (aproximadamente 300.000 km/s), a longa distância de ida e volta introduz um atraso substancial para os satélites GEO. Dependendo da distância entre o usuário e a estação terrestre, e também da elevação do satélite acima do horizonte, o tempo total de trânsito está entre 250 e 300 ms. Um valor típico é 270 ms (540 ms, no caso de um sistema VSAT com um hub).

Para fins de comparação, os enlaces de micro-ondas terrestres têm um atraso de propagação de aproximadamente 3 μs/km, e os enlaces de cabo coaxial ou fibra óptica geram um atraso de cerca de 5 μs/km. Neste último caso, o atraso é maior porque os sinais eletromagnéticos trafegam com maior rapidez no ar que em materiais sólidos.

Outra propriedade importante dos satélites é que eles basicamente são meios de difusão. Enviar uma mensagem para milhares de estações localizadas na área de cobertura de um transponder não custa mais do que enviar a mensagem para apenas uma estação. Para algumas aplicações, essa propriedade é muito útil. Por exemplo, poderíamos imaginar um satélite transmitindo páginas da Web comuns para os caches de um grande número de computadores espalhados por uma extensa área. Mesmo quando o broadcasting pode ser simulado com o uso de linhas ponto a ponto, o broadcasting por satélite pode ser muito mais econômico. Por outro lado, do ponto de vista da segurança e da privacidade, os satélites são um completo desastre: todo mundo pode ouvir tudo. A criptografia é essencial quando é necessário segurança.

Nos satélites, o custo de transmissão de uma mensagem é independente da distância percorrida. O serviço de uma chamada transcontinental não custa mais do que uma chamada entre um lado e outro da rua. Os satélites também proporcionam excelentes taxas de erros e podem ser implementados quase instantaneamente, um detalhe fundamental para a comunicação militar.

2.4.2 Satélites terrestres de órbita média

Em altitudes muito mais baixas, entre os dois cinturões de Van Allen, encontramos os **satélites de órbita média**, ou **MEO** (**Medium-Earth Orbit**). Vistos da Terra, esses satélites se deslocam lentamente em longitude, levando cerca de seis horas para circular a Terra. Consequentemente, eles devem ser acompanhados à medida que se movem pelo céu. Pelo fato de estarem em órbitas mais baixas que os GEOs, eles têm uma área de cobertura menor no solo e exigem transmissores menos potentes para alcançá-los. Atualmente, esses satélites não são usados para telecomunicações, portanto, não os examinaremos mais aqui. Os trinta satélites **GPS** (**Global Positioning System**) que estão em órbita a cerca de 20.200 km de altitude são exemplos de satélites MEO.

2.4.3 Satélites terrestres de órbita baixa

A uma altitude menor, encontramos os **satélites de órbita baixa**, ou **LEO** (**Low-Earth Orbit**). Em razão de seu rápido movimento, são necessárias grandes quantidades desses satélites para formar um sistema completo. Por outro lado, pelo fato de os satélites estarem muito próximos da Terra, as estações terrestres não precisam de muita potência, e o atraso de ida e volta é de apenas alguns milissegundos. O custo de lançamento também é muito mais baixo. Nesta seção, examinaremos dois exemplos de constelações de satélites para o serviço de voz, Iridium e Globalstar.

Durante os primeiros trinta anos da era do satélite, os satélites de baixa órbita raramente eram usados, porque apareciam e desapareciam de vista com muita rapidez. Em 1990, a Motorola deu início a um novo empreendimento e enviou um requerimento à FCC, solicitando permissão para lançar 77 satélites de baixa órbita para o projeto **Iridium** (o elemento 77 é o irídio). Mais tarde, o plano foi revisto para que fossem usados apenas 66 satélites; assim, o projeto deveria ter seu nome alterado para Dysprosium (o elemento 66), mas esse nome provavelmente lembrava muito mais uma doença do que um satélite. A ideia era que, assim que um satélite estivesse fora de vista, outro o substituiria. Essa proposta criou uma agitação entre outras empresas de comunicações. De repente, todas elas quiseram lançar uma cadeia de satélites de baixa órbita.

Após sete anos reunindo parceiros e financiamentos, o serviço de comunicação se iniciou em novembro de 1998. Infelizmente, a demanda comercial por grandes e pesados telefones via satélite era desprezível, porque a rede de telefonia móvel (celular) havia crescido de modo espetacular desde 1990. Como consequência, o Iridium não gerou lucro e foi à falência em agosto de 1999, em um dos mais espetaculares fiascos corporativos da história. Os satélites e outros bens (no valor de 5 bilhões de dólares) foram ad-

quiridos mais tarde por um investidor por 25 milhões de dólares, em uma espécie de venda de garagem extraterrestre. Outros empreendimentos comerciais logo se seguiram.

O serviço Iridium foi reiniciado em março de 2001, e tem crescido desde então. Há serviços de voz, dados, busca, fax e navegação em qualquer lugar do mundo, seja em terra, seja em mar e ar, com dispositivos portáteis que se comunicam diretamente com os satélites Iridium. Os clientes incluem as indústrias marítima, de aviação e de exploração de petróleo, bem como pessoas que viajam para regiões do mundo que não têm uma infraestrutura de telecomunicações (por exemplo, desertos, montanhas, o Polo Sul e alguns países do Terceiro Mundo).

Os satélites Iridium estão posicionados a uma altitude de 750 km, em órbitas polares circulares. Eles estão organizados em eixos norte-sul, com um satélite a cada 32 graus de latitude, conforme mostra a Figura 2.15. Cada satélite tem no máximo 48 células (feixes pontuais), com uma capacidade de 3.840 canais, alguns deles usados para busca e navegação, enquanto outros são empregados para dados e voz.

Com seis eixos de satélite, a Terra inteira é coberta, como sugere a Figura 2.15. Uma propriedade interessante do Iridium é que a comunicação entre clientes distantes ocorre no espaço, como ilustra a Figura 2.16(a). Na figura, vemos um chamador no Polo Norte entrando em contato com um satélite situado diretamente acima dele. Cada satélite tem quatro vizinhos com os quais pode se comunicar, dois no mesmo eixo (mostrado) e dois em eixos adjacentes (não mostrados). Os satélites repassam a chamada por essa grade até finalmente seja enviado para o destinatário no Polo Sul.

Um projeto alternativo para o Iridium é o **Globalstar**. Ele se baseia em 48 satélites LEO, mas utiliza um esquema de comutação diferente do que é usado no Iridium. Enquanto este retransmite as chamadas de satélite para satélite, o que exige sofisticado equipamento de comutação nos satélites, o Globalstar utiliza um projeto tradicional de canal em curva. A chamada originada no Polo Norte na Figura 2.16(b) é enviada de volta à Terra e recebida pela grande estação terrestre na fábrica de brinquedos do Papai Noel. A chamada é, então, roteada por uma rede terrestre até a estação terrestre mais próxima ao destino, e é entregue por uma conexão de canal em curva da maneira ilustrada. A vantagem desse esquema é que ele coloca a maior parte da complexidade no solo, onde é mais fácil de administrar. Além disso, o uso de grandes antenas nas estações terrestres, capazes de emitir um sinal potente e receber um sinal fraco, significa que podem ser utilizados telefones de potência mais baixa. Afinal, o telefone emite apenas alguns miliwatts de potência e, assim, o sinal que volta para a estação terrestre é bastante fraco, mesmo depois de ter sido amplificado pelo satélite.

Figura 2.15 | Os satélites Iridium formam seis eixos em torno da Terra.

Os satélites continuam a ser lançados a uma taxa de algo em torno de 20 por ano, incluindo satélites cada vez maiores, que agora pesam mais de 5.000 kg. Mas há também satélites muito pequenos para organizações mais preocupadas com o orçamento. Para tornar a pesquisa do espaço mais acessível, os acadêmicos de Cal Poly e Stanford se reuniram em 1999 para definir um padrão para satélites em miniatura e um disparador associado que reduziria bastante os custos de lançamento (Nugent et al., 2008). **CubeSats** são satélites em unidades de cubos de 10 cm · 10 cm · 10 cm, cada um pesando menos de 1 kg, que podem ser lançados a partir de US$ 40.000 cada. O disparador voa como um segundo payload nas missões espaciais comerciais. Ele é basicamente um tubo que ocupa três unidades de cubesats e usa molas para lançá-los em órbita. Cerca de 20 cubesats foram lançados até agora, com muito mais em andamento. A maioria deles se comunica com estações terrestres nas faixas de UHF e VHF.

2.4.4 Comparação entre satélites e fibra óptica

Uma comparação entre as comunicações por satélite e terrestre é instrutiva. Há 25 anos, pensava-se que o futuro da comunicação residia nos satélites de comunicações. Afinal, o sistema telefônico mudou muito pouco nos últimos cem anos e não mostrou sinais de mudança para os próximos cem. Esse movimento glacial foi causado, em grande parte, pelo ambiente regulador no qual se esperava que as companhias telefônicas fornecessem bons serviços de voz a preços razoáveis (o que elas fizeram) e, em troca, tivessem lucro garantido sobre seu investimento. Havia modems de 1.200 bps disponíveis para as pessoas que precisavam transmitir dados. Isso era praticamente tudo o que existia na época.

Com o surgimento da concorrência, em 1984 nos Estados Unidos e um pouco mais tarde na Europa, esse quadro se alterou radicalmente. As companhias telefônicas começaram a substituir suas redes de longa distância por fibra óptica e introduziram serviços de alta largura de banda, como ADSL. Essas empresas também interromperam sua

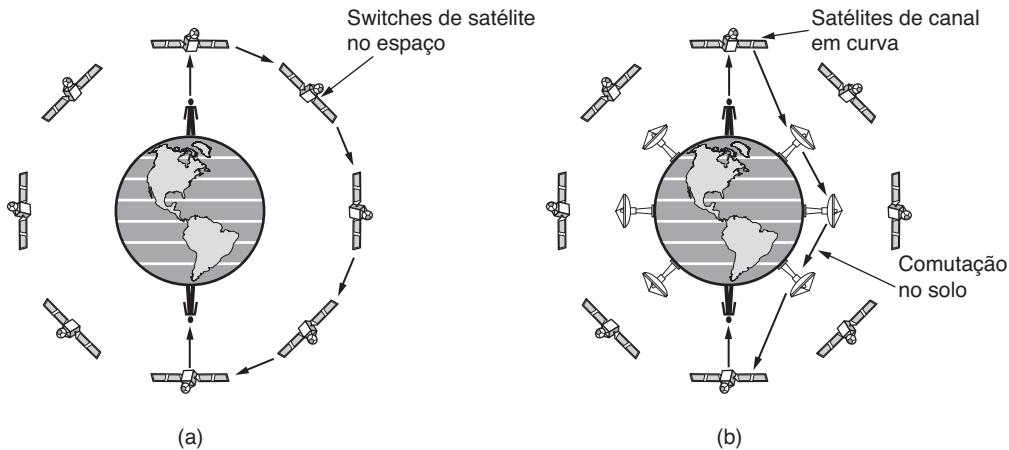



Figura 2.16 | (a) Retransmissão no espaço. (b) Retransmissão no solo.

antiga prática de cobrar preços artificialmente elevados a usuários de serviços de longa distância, a fim de subsidiar o serviço local. Subitamente, as conexões terrestres de fibra pareciam ser a melhor opção a longo prazo.

Apesar disso, os satélites de comunicações têm alguns segmentos de mercado muito importantes, que a fibra óptica não é capaz de alcançar. Primeiro, quando a implantação rápida é crítica, os satélites ganham facilmente. Uma resposta rápida é útil para sistemas de comunicação militares em tempos de guerra e resposta a desastres em tempos de paz. Em Sumatra, em dezembro de 2004, após o grande terremoto e o tsunami subsequente, por exemplo, os satélites de comunicações foram capazes de restaurar as comunicações com os primeiros respondedores dentro de 24 horas. Essa resposta rápida foi possível porque existe um mercado de serviço de satélite desenvolvido em que grandes participantes, como Intelsat, com mais de 50 satélites, podem arrendar capacidade onde quer que ela seja necessária. Para clientes atendidos por redes de satélite existentes, um VSAT pode ser preparado fácil e rapidamente para fornecer um enlace de megabits/s para qualquer parte do mundo.

Um segundo nicho de mercado ocorre em lugares onde a infraestrutura terrestre é pouco desenvolvida. Muitas pessoas hoje em dia querem se comunicar enquanto se movimentam. As redes de telefone móvel abrangem locais com boa densidade populacional, mas não realizam um trabalho adequado em outros lugares (por exemplo, no mar ou no deserto). Ao contrário, o Iridium oferece serviço de voz em qualquer lugar do planeta, até mesmo no Polo Sul. A infraestrutura em terra pode ser cara para instalar, dependendo do terreno e dos direitos necessários para viabilizar o meio. A Indonésia, por exemplo, tem seu próprio satélite para o tráfego de telefone doméstico. Lançar um satélite foi mais barato do que esticar milhares de cabos submarinos entre as 13.677 ilhas do arquipélago.

Um terceiro nicho se relaciona a situações em que a difusão é essencial. Uma mensagem enviada por satélite pode ser recebida por milhares de estações terrestres ao mesmo tempo. Os satélites são usados para distribuir grande parte da programação da TV para estações locais por esse motivo. Agora, existe um grande mercado para transmissões de rádio e TV por satélite, diretamente para usuários finais com receptores de satélite em suas casas e carros. Vários outros tipos de conteúdo também podem ser transmitidos. Por exemplo, uma empresa que transmite um fluxo de preços de ações, apólices ou mercadorias a milhares de corretores deve considerar que um sistema de satélite é mais econômico que simular a difusão no solo.

Resumindo, parece que a comunicação do futuro será feita por fibras ópticas terrestres combinadas com rádio celular, mas, para algumas aplicações específicas, os satélites são melhores. Entretanto, existe um motivo que se aplica a tudo isso: a economia. Embora a fibra ofereça mais largura de banda, é muito provável que a comunicação terrestre e por satélite entre em uma concorrência agressiva por melhores preços. Se os avanços tecnológicos reduzirem radicalmente o custo de exploração de um satélite (por exemplo, se no futuro algum veículo espacial puder lançar dezenas de satélites de uma só vez), ou se os satélites de baixa órbita se desenvolverem, não é certo que a fibra vencerá em todos os mercados.

2.5 Modulação digital e multiplexação

Agora que já estudamos as propriedades dos canais com e sem fios, voltamos nossa atenção para o problema de enviar informações digitais. Os canais com fio e sem fios transportam sinais analógicos, como a tensão variando continuamente, a intensidade de luz ou a intensidade de som. Para enviar informações digitais, temos de criar sinais

digitais para representar os bits. O processo de conversão entre bits e sinais que os representam é chamado **modulação digital**.

Começaremos com esquemas que convertem diretamente os bits em um sinal. Esses esquemas resultam em **transmissão de banda base**, em que o sinal ocupa frequências de zero até um máximo que depende da taxa de sinalização. Ele é comum para fios. Depois, vamos considerar os esquemas que regulam a amplitude, a fase ou a frequência de um sinal da portadora para transportar bits. Esses esquemas resultam em **transmissão de banda passante**, em que o sinal ocupa uma banda de frequências em torno da frequência do sinal da portadora. Isso é comum para canais sem fio e ópticos, para os quais os sinais devem residir em uma determinada banda de frequência.

Os canais normalmente são compartilhados por vários sinais. Afinal, é muito mais conveniente usar um único fio para transportar vários sinais do que instalar um fio para cada sinal. Esse tipo de compartilhamento é chamado **multiplexação**. Ele pode ser realizado de diversas maneiras. Apresentaremos métodos para multiplexação por divisão de tempo, frequência e código.

As técnicas de modulação e multiplexação que descrevemos nesta seção são todas bastante usadas para canais com fios, de fibra, terrestres sem fios e por satélite. Nas próximas seções, examinaremos exemplos de redes para vê-las em ação.

2.5.1 Transmissão em banda base

A forma mais simples de modulação digital é usar uma tensão positiva para representar 1 e uma tensão negativa para representar 0. Para uma fibra óptica, a presença de luz poderia representar 1 e a ausência de luz, 0. Esse esquema é chamado **NRZ** (**Non-Return-to-Zero**). O nome esquisito tem motivos históricos, e significa simplesmente que o sinal acompanha os dados. Um exemplo aparece na Figura 2.17(b).

Uma vez enviado, o sinal NRZ se propaga pelo fio. Na outra ponta, o receptor o converte para bits fazendo a amostragem do sinal em intervalos de tempo regulares. Esse sinal não ficará exatamente como o enviado. Ele será atenuado e distorcido pelo canal e pelo ruído no receptor. Para decodificar os bits, o receptor mapeia as amostras de sinal para os símbolos mais próximos. Para o NRZ, uma tensão positiva será considerada para indicar que foi enviado um bit 1, e uma tensão negativa será considerada como indicação de que foi enviado um 0.

O NRZ é um bom ponto de partida para nossos estudos, pois é simples, mas na prática raramente é usado por si só. Esquemas mais complexos podem converter bits em sinais, os quais atendem melhor às considerações da engenharia. Esses esquemas são chamados **códigos de linha**. A seguir, descreveremos os códigos de linha que ajudam com eficiência da largura de banda, recuperação do clock e balanço do componente de CC.

Eficiência da largura de banda

Com o NRZ, o sinal pode alternar entre os níveis positivo e negativo até a cada 2 bits (no caso de 1s e 0s alternados). Isso significa que precisamos de uma largura de banda de pelo menos $B/2$ Hz quando a taxa de bits é B bits/s. Essa relação vem da taxa de Nyquist (Equação 2.2). Por ser este um limite fundamental, não podemos usar o NRZ mais rápido sem usar mais largura de banda. Esta normalmente é um recurso limitado, até mesmo para redes com fios. Sinais de maior frequência são cada vez mais atenuados, tornando-os menos úteis, e sinais de maior frequência também exigem circuitos eletrônicos mais rápidos.

Uma estratégia para usar largura de banda limitada com mais eficiência é usar mais de dois níveis de sinaliza-

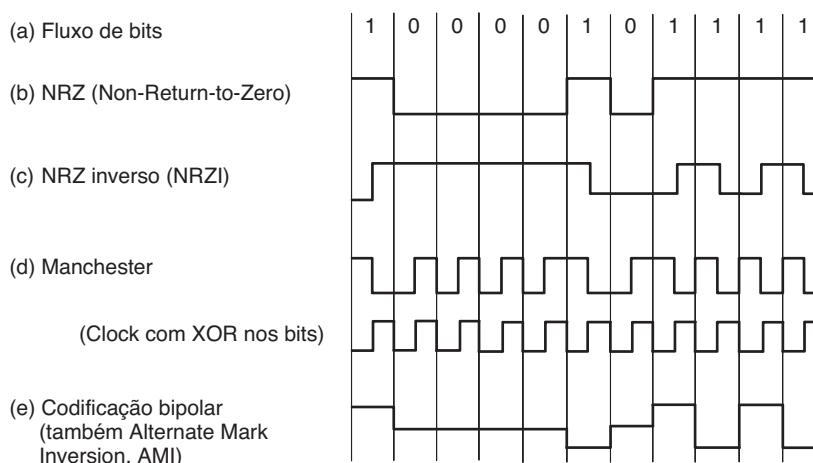



Figura 2.17 | Códigos de linha: (a) Bits, (b) NRZ, (c) NRZI, (d) Manchester, (e) Bipolar or AMI.

ção. Usando quatro voltagens, por exemplo, podemos enviar 2 bits ao mesmo tempo como um único **símbolo**. Esse projeto funcionará desde que o sinal no receptor seja suficientemente forte para distinguir os quatro níveis. A taxa em que o sinal muda é, então, metade da taxa de bits, de modo que a largura de banda necessária foi reduzida.

Chamamos a taxa em que o sinal muda de **taxa de símbolos**, para distingui-la da **taxa de bits**. Esta é a taxa de símbolos multiplicada pelo número de bits por símbolo. Um nome mais antigo para a taxa de símbolos, principalmente no contexto de dispositivos chamados modems de telefone que transmitem dados digitais pelas linhas telefônicas, é **taxa baud**. Na literatura, os termos 'taxa de bits' e 'taxa baud' normalmente são usados de modo incorreto.

Observe que o número de níveis de sinal não precisa ser uma potência de dois. Normalmente ele não é, com alguns dos níveis usados para proteção contra erros e simplificação do projeto do receptor.

Recuperação de clock

Para todos os esquemas que codificam bits em símbolos, o receptor precisa saber quando um símbolo termina e o próximo começa, para decodificar os bits corretamente. Com o NRZ, em que os símbolos são simplesmente níveis de tensão, uma longa sequência de 0s ou 1s deixa o sinal inalterado. Após um tempo, é difícil distinguir os bits, pois 15 zeros se parecem muito com 16 zeros, a menos que você tenha um clock muito preciso.

Os clocks precisos ajudariam com esse problema, mas eles são uma solução dispendiosa para equipamentos básicos. Lembre-se de que estamos temporizando bits em links que trabalham em muitos megabits/s, de modo que o clock teria de fluir por menos de uma fração de microssegundo pela maior sequência permitida. Isso pode ser razoável para enlaces lentos ou mensagens curtas, mas essa não é uma solução geral.

Uma estratégia é enviar um sinal de clock separado para o receptor. Outra linha de clock não é grande coisa para barramentos de computador ou cabos curtos, em que existem muitas linhas paralelas, mas é um desperdício para a maioria dos enlaces de rede, pois, se tivéssemos outra linha para enviar um sinal, poderíamos usá-la para enviar dados. Aqui, um truque inteligente é misturar o sinal de clock com o sinal de dados efetuando a operação XOR por ambos, de modo que nenhuma linha extra seja necessária. Os resultados aparecem na Figura 2.17(d). O clock faz uma transição de clock em cada tempo de bit, de modo que ele funciona no dobro da taxa de bits. Quando ele é XORado com o nível 0, ele faz a transição de baixo para alto, que é simplesmente o clock. Essa transição é o 0 lógico. Quando ele é XORado com o nível 1, é invertido e faz uma transição de alto para baixo. Essa transição é o 1 lógico. Esse esquema é conhecido como codificação **Manchester**, e foi usado para a Ethernet clássica.

A desvantagem da codificação Manchester é que ela exige duas vezes mais largura de banda que a NRZ, em virtude do clock, e aprendemos que a largura de banda normalmente importa. Uma estratégia diferente é baseada na ideia de que devemos codificar os dados para garantir que haja transições suficientes no sinal. Considere que o NRZ terá problemas de recuperação de clock somente para longas sequências de 0s e 1s. Se houver transições frequentes, será fácil para o receptor permanecer sincronizado com o fluxo de símbolos de chegada.

Como um passo na direção certa, podemos simplificar a situação codificando um 1 como uma transição e 0 como nenhuma transição, ou vice-versa. Essa codificação é chamada **NRZI** (**Non-Return-to-Zero Inverted**), uma variação da NRZ. Um exemplo pode ser visto na Figura 2.17(c). O popular padrão **USB** (**Universal Serial Bus**) para a conexão de periféricos de computador usa NRZI. Com ele, longas sequências de 1s não causam problema.

Naturalmente, longas sequências de 0s ainda causam um problema que precisamos consertar. Se fôssemos a companhia telefônica, poderíamos simplesmente solicitar que o transmissor não transmitisse muitos 0s. Para que funcionassem corretamente, as linhas telefônicas digitais mais antigas nos Estados Unidos, chamadas **linhas T1**, realmente exigiam que não mais do que 15 zeros consecutivos fossem enviados. Para resolver realmente o problema, podemos dividir sequências de 0s mapeando pequenos grupos de bits para serem transmitidos de modo que os grupos com 0s sucessivos sejam mapeados para padrões ligeiramente maiores, que não têm muitos 0s consecutivos.

Um código bem conhecido para fazer isso é chamado **4B/5B**. Cada 4 bits são mapeados para um padrão de 5 bits com uma tabela de tradução fixa. Os padrões de 5 bits são escolhidos de tal forma que nunca haverá uma sequência de mais de três 0s consecutivos. O mapeamento aparece na Tabela 2.4. Esse esquema acrescenta 25 por cento de overhead, que é melhor do que os 100 por cento de overhead da codificação Manchester. Como existem 16 combinações de entrada e 32 combinações de saída, algumas das combinações de saída não são usadas. Deixando de lado as combinações com muitos 0s sucessivos, ainda existirão alguns códigos. Como um bônus, podemos usar esses códigos não de dados para representar sinais de controle da camada física. Por exemplo, em alguns casos, '11111' representa a linha ociosa, e '11000' representa o início de um quadro.

Uma técnica alternativa é fazer com que os dados pareçam aleatórios, o que é conhecido como embaralhamento (ou scrambling). Nesse caso, provavelmente haverá transições frequentes. Um **embaralhador** funciona realizando o XOR dos dados com uma sequência pseudoaleatória antes de serem transmitidos. Esse embaralhamento tornará os dados tão aleatórios quanto a sequência pseudoaleatória (supondo que eles sejam independentes da sequência

pseudoaleatória). O receptor, então, realiza o XOR dos bits de entrada com a mesma sequência pseudoaleatória para registrar os dados reais. Para que isso seja prático, a sequência pseudoaleatória precisa ser fácil de criar. Em geral, ela é dada como uma semente para um gerador de números aleatórios simples.

O embaralhamento é atraente porque não acrescenta largura de banda ou overhead temporal. De fato, normalmente é importante condicionar o sinal de modo que ele não mantenha energia em componentes de frequência dominantes (causados por padrões de dados repetitivos) que poderiam radiar interferência eletromagnética. O embaralhamento ajuda porque os sinais aleatórios tendem a ser 'brancos' ou ter a energia espalhada pelos demais componentes de frequência.

Contudo, o embaralhamento não garante que não haverá sequências longas. Ocasionalmente, é possível não ter sorte. Se os dados forem iguais à sequência pseudoaleatória, com o XOR, eles serão todos 0. Esse resultado geralmente não ocorre com uma sequência pseudoaleatória longa, difícil de prever. Contudo, com uma sequência curta ou previsível, é possível que usuários maliciosos enviem padrões de bits que causem longas sequências de 0s depois do embaralhamento, causando falha nos enlaces. As antigas versões dos padrões para enviar pacotes IP por enlaces SONET no sistema telefônico tinham esse defeito (Malis e Simpson, 1999). Era possível que os usuários enviassem certos 'pacotes fatais', sabendo que isso causaria problemas.

Sinais balanceados

Sinais que possuem tanto tensão positiva quanto negativa, mesmo que seja por curtos períodos, são chamados **sinais balanceados**. Sua média é zero, o que significa que eles não possuem o componente elétrico CC (corrente contínua). A falta de um componente CC é uma vantagem, pois alguns canais, como o cabo coaxial ou as linhas com transformadores, atenuam bastante um componente CC, em virtude de suas propriedades físicas. Além disso, o método de conexão do receptor ao canal de comunicação, chamado **acoplamento capacitivo**, deixa passar apenas a parte CA (corrente alternada) de um sinal. De qualquer modo, se enviarmos um sinal cuja média não seja zero, desperdiçamos energia, pois o componente CC será eliminado.

Dados (4B)	Código (5B)	Dados (4B)	Código (5B)
0000	11110	1000	10010
0001	01001	1001	10011
0010	10100	1010	10110
0011	10101	1011	10111
0100	01010	1100	11010
0101	01011	1101	11011
0110	01110	1110	11100
0111	01111	1111	11101

Tabela 2.4 | Mapeamento 4B/5B.

O balanceamento ajuda a oferecer transições para a recuperação de clock, pois existe uma mistura de tensões positiva e negativa. Ele também oferece um modo simples de calibrar receptores, pois a média do sinal pode ser medida e usada como um patamar de decisão para decodificar símbolos. Com sinais não balanceados, a média pode flutuar do nível de decisão verdadeiro, em razão de uma alta taxa de '1s', por exemplo, que faria com que mais símbolos fossem decodificados com erros.

Um modo simples de construir um código balanceado é usar dois níveis de tensão para representar um '1' lógico (digamos, +1 V ou –1 V), com 0 V representando um zero lógico. Para enviar 1, o transmissor alterna entre os níveis +1 V e –1 V, de modo que eles sempre resultem na média. Esse esquema é chamado **codificação bipolar**. Nas redes de telefone, ele é chamado **AMI** (**Alternate Mark Inversion**), baseado em uma terminologia antiga, em que 1 é chamado 'marca' e 0 é chamado 'espaço' Um exemplo pode ser visto na Figura 2.17(e).

A codificação bipolar acrescenta um nível de tensão para alcançar o equilíbrio. Como alternativa, podemos usar um mapeamento como 4B/5B para conseguir o equilíbrio (bem como as transições para a recuperação de clock). Um exemplo desse tipo de código balanceado é o código de linha **8B/10B**. Ele mapeia 8 bits de entrada para 10 bits de saída, de modo que é 80 por cento eficiente, assim como o código de linha 4B/5B. Os 8 bits são divididos em um grupo de 5 bits, que é mapeado para 6 bits, e um grupo de 3 bits, que é mapeado para 4 bits. Os símbolos de 6 e 4 bits são, então, concatenados. Em cada grupo, alguns padrões de entrada podem ser mapeados para padrões de saída balanceados, com o mesmo número de 0s e 1s. Por exemplo, '001' é mapeado para '1001', que é balanceado. Mas não existem combinações suficientes para todos os padrões de saída serem balanceados. Para esses casos, cada padrão de entrada é mapeado para dois padrões de saída. Um terá um 1 extra e a alternativa terá um 0 extra. Por exemplo, '000' é mapeado tanto para '1011' quanto para seu complemento '0100'. À medida que os bits de entrada são mapeados para bits de saída, o codificador recorda a **disparidade** do símbolo anterior. A disparidade é o número total de 0s ou 1s pelos quais o sinal está fora de equilíbrio. O codificador, então, seleciona um padrão de saída ou seu padrão alternativo, para reduzir a disparidade. Com 8B/10B, a disparidade será de no máximo 2 bits. Assim, o sinal nunca estará longe de ser balanceado. Nunca haverá mais do que cinco 1s ou 0s consecutivos, para ajudar com a recuperação do clock.

2.5.2 Transmissão em banda passante

Em geral, queremos usar uma faixa de frequências que não começa com zero para enviar informações por um canal. Para canais sem fio, não é prático enviar sinais de frequência muito baixos, pois o tamanho da antena precisa ser uma fração do comprimento de onda do sinal, que se torna grande. De qualquer forma, restrições da regulamentação e a necessidade de evitar interferência normalmente ditam a escolha de frequências. Até mesmo para fios, colocar um sinal em determinada banda de frequência é útil para permitir que diferentes tipos de sinais coexistam no canal. Esse tipo de transmissão é chamado transmissão de banda passante, pois uma banda de frequência arbitrária é usada para passar o sinal.

Felizmente, nossos resultados fundamentais, mostrados anteriormente no capítulo, são todos em termos de largura de banda, ou da largura da banda de frequência. Os valores de frequência absoluta não importam para a capacidade. Isso significa que podemos apanhar um sinal de **banda base**, que ocupa de 0 a B Hz, e deslocá-lo para cima para ocupar uma **banda passante** de S a $S + B$ Hz sem mudar a quantidade de informação que ele pode transportar, embora o sinal tenha aparência diferente. Para processar um sinal no receptor, podemos deslocá-lo de volta para a banda base, onde é mais conveniente detectar símbolos.

A modulação digital é realizada com a transmissão da banda passante regulando ou modulando um sinal de portadora sobreposto à banda passante. Podemos modular a amplitude, a frequência ou a fase do sinal da portadora. Cada um desses métodos tem um nome correspondente. No **ASK** (**Amplitude Shift Keying**), duas amplitudes diferentes são usadas para representar 0 e 1. Um exemplo com um nível diferente de zero e outro zero aparece na Figura 2.18(b). Mais de dois níveis podem ser usados para representar mais símbolos. De modo semelhante, com o chaveamento por deslocamento de frequência, ou **FSK** (**Frequency Shift Keying**), dois ou mais tons diferentes são usados. O exemplo na Figura 2.18(c) usa apenas duas frequências. Na forma mais simples do chaveamento por deslocamento de fase, ou **PSK** (**Phase Shift Keying**), a onda da portadora é sistematicamente deslocada em 0 ou 180 graus em cada período de símbolo. Como existem duas fases, ela é chamada **chaveamento binário por deslocamento de fase**, ou **BPSK** (**Binary Phase Shift Keying**). O chaveamento 'binário', nesse caso, refere-se aos dois símbolos, não que os símbolos representem 2 bits. Um exemplo aparece na Figura 2.18(c). Um esquema melhor, que usa a largura de banda do canal de modo mais eficiente, é usar quatro deslocamentos, por exemplo, 45, 135, 225 ou 315 graus, para transmitir 2 bits de informação por símbolo. Essa versão é chamada **chaveamento por deslocamento de fase em quadratura**, ou **QPSK** (**Quadrature Phase Shift Keying**).

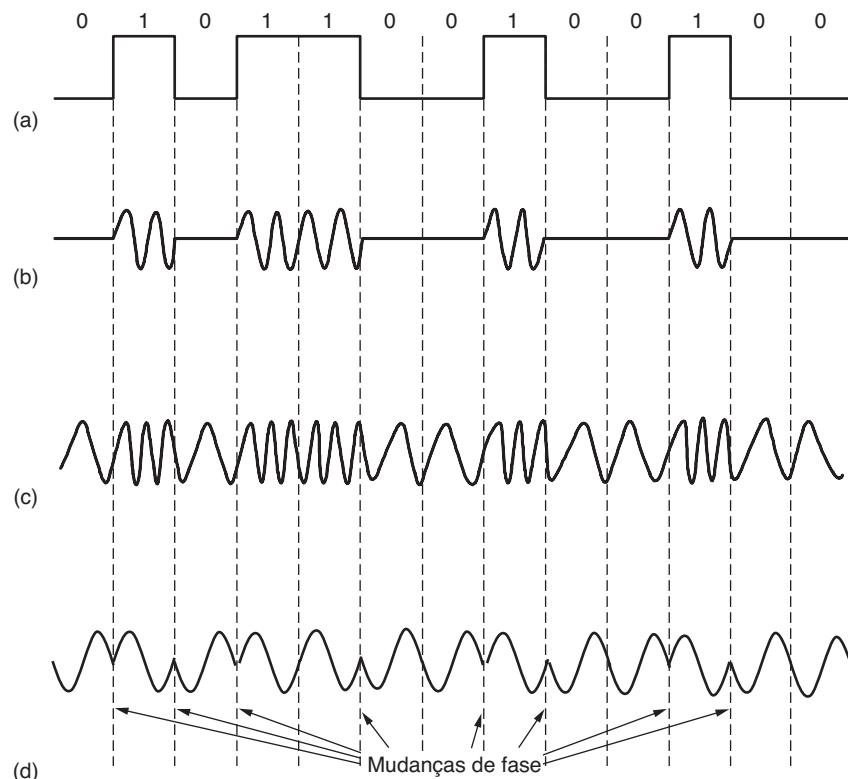



Figura 2.18 | (a) Um sinal binário. (b) Chaveamento por deslocamento de amplitude. (c) Chaveamento por deslocamento de frequência. (d) Chaveamento por deslocamento de fase.

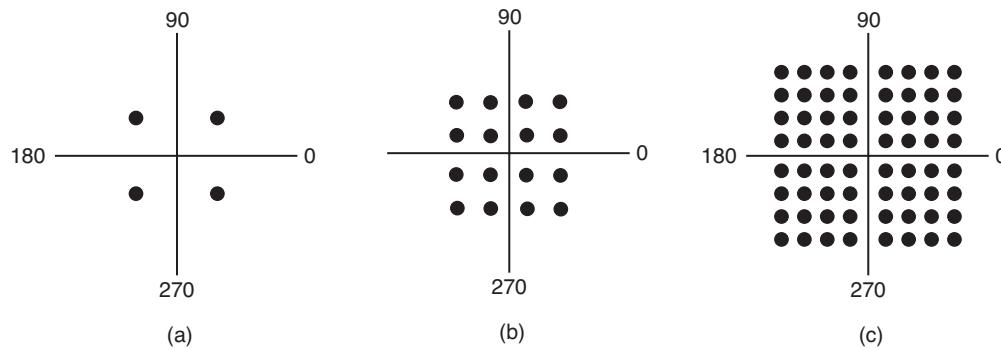



Figura 2.19 (a) QPSK. (b) QAM-16. (c) QAM-64.

Podemos combinar esses esquemas e usar mais níveis para transmitir mais bits por símbolo. Somente a frequência ou a fase podem ser moduladas em determinado momento, pois elas estão relacionadas, normalmente sendo a taxa de mudança de fase com o tempo. Em geral, amplitude e fase são moduladas em combinação. Três exemplos aparecem na Figura 2.19. Em cada exemplo, os pontos dão as combinações válidas de amplitude e de fase de cada símbolo. Na Figura 2.19(a), vemos pontos equidistantes em 45, 135, 225 e 315 graus. A fase de um ponto é indicada pelo ângulo que uma linha a partir dele até a origem faz com o eixo x positivo. A amplitude de um ponto é a distância a partir da origem. Essa figura é uma representação do QPSK.

Esse tipo de diagrama é chamado **diagrama de constelação**. Na Figura 2.19(b), vemos um esquema de modulação com um diagrama de constelação mais denso. Como são utilizadas dezesseis combinações de amplitudes e fase, o esquema de modulação pode ser usado para transmitir quatro bits por símbolo. Ele é chamado **QAM-16**, em que QAM significa **Quadrature Amplitude Modulation** (**modulação por amplitude de quadratura**). A Figura 2.19(c) é outro esquema de modulação com 64 combinações diferentes, de forma que podem ser transmitidos 6 bits por símbolo. Ele é chamado **QAM-64**. Também são usadas QAMs de ordem mais alta. Como você poderia imaginar a partir dessas constelações, é mais fácil criar circuitos eletrônicos para produzir símbolos como uma combinação de valores em cada eixo do que como uma combinação de valores de amplitude e fase. É por isso que os padrões se parecem com quadrados, em vez de círculos concêntricos.

As constelações que vimos até aqui não mostram de forma alguma como os bits são atribuídos aos símbolos. Ao fazer a atribuição, uma consideração importante é que uma pequena rajada de ruído no receptor não ocasiona muitos erros de bit. Isso poderia acontecer se atribuíssemos valores de bit consecutivos a símbolos adjacentes. Com a QAM-16, por exemplo, se um símbolo designasse 0111 e o símbolo vizinho designasse 1000, se o receptor por engano apanhasse o símbolo adjacente, ele faria com que todos os bits ficassem errados. Uma solução melhor é mapear entre bits e símbolos, de modo que os símbolos adjacentes difiram em apenas 1 posição de bit. Esse mapeamento é chamado **código Gray**. A Figura 2.20 mostra um diagrama de constelação QAM-16 que foi codificada pelo código Gray. Agora, se o receptor decodificar o símbolo com erro, ele cometerá apenas um erro de único bit no caso esperado em que o símbolo codificado está próximo do símbolo transmitido.

2.5.3 Multiplexação por divisão de frequência

Os esquemas de modulação que vimos nos permitem enviar um sinal para transmitir bits por um enlace com ou sem fios. Porém, as economias de escala desempenham um papel importante no modo como usamos as redes. Custa basicamente o mesmo valor instalar e manter uma linha de transmissão com uma largura de banda alta e uma linha com largura de banda baixa entre dois escritórios diferentes (em outras palavras, os custos advêm de ter de cavar a trincheira, não do tipo de cabo ou fibra que se utiliza). Consequentemente, esquemas de multiplexação têm sido desenvolvidos para compartilhar as linhas entre muitos sinais.

A **multiplexação por divisão de frequência**, ou **FDM** (**Frequency Division Multiplexing**), tira proveito da transmissão de banda passante para compartilhar um canal. Ela divide o espectro em bandas de frequência, com cada usuário tendo posse exclusiva de alguma banda para enviar seu sinal. A transmissão de rádio AM ilustra a FDM. O espectro alocado é de cerca de 1 MHz, aproximadamente 500 a 1500 kHz. Diferentes frequências são alocadas a diferentes canais lógicos (estações), cada um operando em uma parte do espectro, com a separação entre canais grande o bastante para impedir interferência.

Para ver um exemplo mais detalhado, a Figura 2.21 mostra como três canais telefônicos de nível de voz são multiplexados com o uso da FDM. Os filtros limitam a largura de banda utilizável a cerca de 3.100 Hz por canal de qualidade de voz. Quando muitos canais são multiplexados ao mesmo tempo, são alocados 4.000 Hz para cada canal. O excesso é chamado **banda de proteção**. Ele mantém os canais bem separados. Primeiro, os canais de voz têm sua

Quando 1101 é enviado:

Ponto	Decodifica como	Erros de bit
A	1101	0
B	1100	1
C	1001	1
D	1111	1
E	0101	1

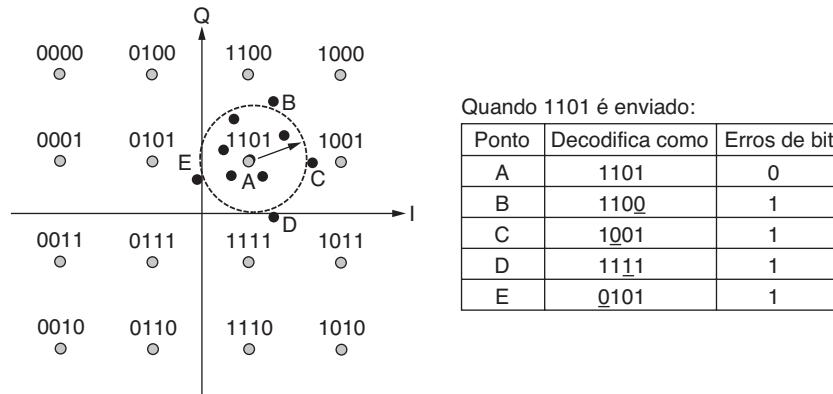



Figura 2.20 | QAM-16 em código Gray.

frequência aumentada, cada qual com um valor diferente. Depois eles podem ser combinados, pois agora não há dois canais ocupando a mesma porção do espectro. Observe que, apesar de haver intervalos entre os canais graças às bandas de proteção, há uma certa sobreposição entre canais adjacentes, porque os filtros não têm limites nítidos. Essa sobreposição significa que um forte pico no limite de um canal será sentido no canal adjacente como ruído não térmico.

Esse esquema tem sido usado para multiplexar chamadas no sistema telefônico há muitos anos, mas o esquema de multiplexação no tempo agora é preferido em lugar dele. Contudo, a FDM continua a ser usada nas redes telefônicas, assim como na telefonia celular, redes sem fios terrestres e redes por satélite em um nível de detalhamento mais alto.

Ao enviar dados digitais, é possível dividir o espectro de modo eficiente sem usar bandas de proteção. Na multiplexação ortogonal por divisão de frequência, ou **OFDM** (**Orthogonal Frequency Division Multiplexing**), a largura de banda do canal é dividida em muitas subportadoras que enviam dados independentemente (por exemplo, com QAM). As subportadoras compactam bastante o domínio de frequência, assim, os sinais de cada subportadora se estendem para as adjacentes. Contudo, como vemos na Figura 2.22, a resposta em frequência de cada subportadora é projetada de modo que seja zero no centro das subportadoras adjacentes. As subportadoras podem, portanto, ser amostradas em suas frequências centrais sem interferência de seus vizinhos. Para que isso funcione, um tempo de proteção é necessário para repetir uma parte dos sinais de símbolo no tempo, de modo que tenham a resposta de

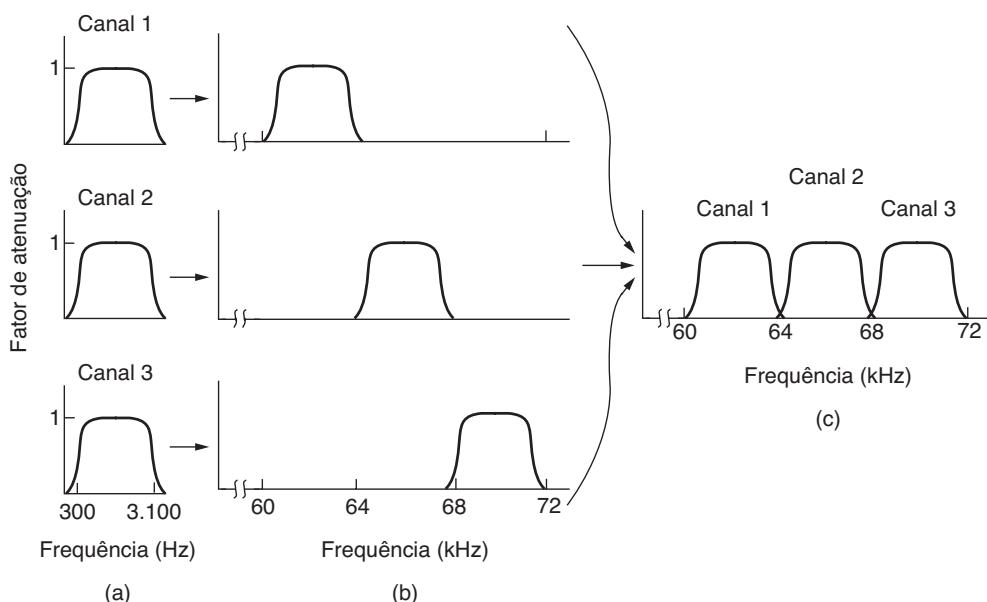



Figura 2.21 | Multiplexação por divisão de frequência. (a) As larguras de banda originais. (b) As larguras de banda aumentaram em frequência. (c) O canal multiplexado.

frequência desejada. Porém, esse overhead é muito menor que o necessário para muitas bandas de proteção.

A ideia da OFDM já existe há muito tempo, mas somente na última década ela foi amplamente adotada, seguindo a observação de que é possível implementar OFDM de modo eficiente em termos de uma transformada de Fourier dos dados digitais por todas as subportadoras (em vez de modular separadamente cada subportadora). A OFDM é usada em 802.11, redes a cabo e redes por linhas de energia elétrica, e é planejada para os sistemas celulares de quarta geração. Normalmente, o fluxo de informações digitais de alta velocidade é dividido em muitos fluxos de baixa velocidade, os quais são transmitidos nas subportadoras em paralelo. Essa divisão é valiosa porque as degradações do canal são mais fáceis de serem lidadas no nível da subportadora; algumas subportadoras podem ser muito degradadas e excluídas em favor de subportadoras que são bem recebidas.

2.5.4 Multiplexação por divisão de tempo

Uma alternativa à FDM é a **multiplexação por divisão de tempo**, ou **TDM** (**Time Division Multiplexing**). Aqui, os usuários se alternam (em um padrão de rodízio), cada um periodicamente usando a largura de banda inteira por um pequeno período. Um exemplo de três fluxos sendo multiplexados com TDM aparece na Figura 2.23. Os bits de cada fluxo de entrada são apanhados em um **slot de tempo** fixo e enviados para o fluxo agregado. Esse fluxo trabalha em uma velocidade que é a soma dos fluxos individuais. Para que isso funcione, os fluxos precisam estar sincronizados no tempo. Pequenos intervalos de **tempo de proteção**, semelhantes à banda de proteção de frequência, podem ser acrescentados para acomodar pequenas variações de sincronização.

A TDM é bastante usada como parte das redes de telefone e celular. Para evitar um ponto de confusão, vamos esclarecer que isso é muito diferente da **multiplexação estatística por divisão de tempo**, ou **STDM** (**Statistical Time Division Multiplexing**). O termo ‘estatística’ é acrescentado para indicar que os fluxos individuais contribuem para o fluxo multiplexado, e *não* sobre uma programação fixa, mas de acordo com a estatística de sua demanda. A STDM é a comutação de pacotes com outro nome.

2.5.5 Multiplexação por divisão de código

Existe um terceiro tipo de multiplexação, que funciona de um modo completamente diferente da FDM e da TDM. A **multiplexação por divisão de código**, ou **CDM** (**Code Division Multiplexing**) é uma forma de comunicação por **dispersão espectral**, na qual um sinal de banda estreita é espalhado por uma banda de frequência mais larga. Isso pode torná-la ainda mais tolerante às interferências, além de permitir que vários sinais de diferentes usuários compartilhem a mesma banda de frequência. Como a multiplexação por divisão de código é usada principalmente para essa última finalidade, ela normalmente é chamada de **acesso múltiplo por divisão de código**, ou **CDMA** (**Code Division Multiple Access**).

O CDMA permite que cada estação transmita por todo o espectro de frequência o tempo todo. Várias transmissões simultâneas são separadas usando a teoria da codificação. Antes de entrarmos no algoritmo, vamos considerar uma analogia: o saguão de um aeroporto com muitos pares de pessoas conversando. Com a TDM, todas as pessoas estariam no meio do saguão, mas conversariam por turnos, um par de pessoas de cada vez. Com a FDM, as pessoas formariam grupos bem separados, cada um mantendo sua própria conversação ao mesmo tempo, alguns com uma altura maior e outros com uma altura menor, de modo que cada par pudesse manter sua própria conversa ao mesmo tempo, mas ainda independentemente dos outros grupos. Com o CDMA, todas as pessoas estariam no meio do saguão falando ao mesmo tempo, mas cada par de pessoas conversando em um idioma diferente. O par que estivesse falando em francês só reconheceria esse idioma, rejeitando tudo que não fosse francês como ruído. Desse modo, a chave do CDMA é a capacidade de extrair o sinal desejado e rejeitar todos os outros como ruído aleatório.

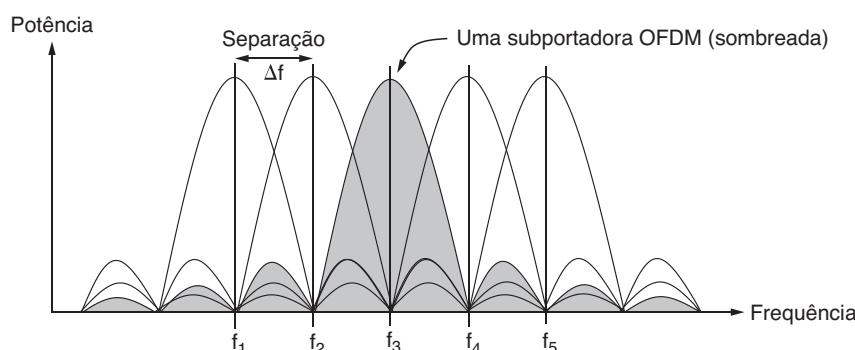



Figura 2.22 | Multiplexação ortogonal por divisão de frequência (OFDM).

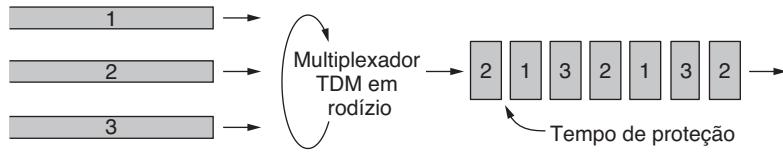



Figura 2.23 | Multiplexação por divisão de tempo (TDM).

A seguir, veremos uma descrição um pouco simplificada do sistema.

No CDMA, cada tempo de duração de um bit é subdividido em *m* intervalos curtos, denominados **chips**. Normalmente, existem 64 ou 128 chips por bit, mas, no exemplo apresentado a seguir, usaremos 8 chips/bit, para simplificar. A cada estação é atribuído um código de *m* bits exclusivo, chamado **sequência de chips**. Para fins pedagógicos, é mais conveniente usar uma notação bipolar para escrever esses códigos como sequências de –1 e +1. Mostraremos as sequências de chips entre parênteses.

Para transmitir um bit 1, uma estação envia sua sequência de chips. Para transmitir um bit 0, envia a negação de sua sequência de chips. Não são permitidos quaisquer outros padrões. Assim, para $m = 8$, se a estação *A* receber a atribuição da sequência de chips (–1 –1 –1 +1 +1 –1 +1 +1), ela transmitirá um bit 1 ao enviar a sequência de chips, e 0 transmitindo (+1 +1 +1 –1 –1 +1 –1 –1). Na realidade, são sinais com esses níveis de tensão que são enviados, mas é suficiente para pensarmos em termos das sequências.

O aumento do volume de informações a serem enviadas de *b* bits/s para *mb* chips/s só poderá ocorrer se a largura de banda disponível for *m* vezes maior que a largura de banda necessária para uma estação que não usa CDMA (supondo-se a ausência de alterações nas técnicas de modulação ou codificação). Se tivéssemos uma banda de 1 MHz disponível para cem estações, com FDM, cada uma teria 10 kHz e poderia transmitir a uma velocidade de 10 kbps (supondo-se 1 bit por Hz). No CDMA, cada estação utiliza 1 MHz inteiro e, portanto, a taxa de chips é de cem chips por bit, para espalhar a taxa de bits da estação de 10 kbps pelo canal.

Na Figura 2.24(a) e (b), mostramos as sequências de chips binárias atribuídas a quatro exemplos de estações e os sinais que elas representam. Cada estação tem sua própria sequência exclusiva de chips. Vamos usar o símbolo **S** para indicar o vetor de *m* chips correspondente à estação *S*, e $\overline{\mathbf{S}}$ para sua negação. Todas as sequências de chips são **ortogonais** par a par; isso significa que o produto interno normalizado de duas sequências de chips distintas, **S** e **T** (indicado como **S • T**), é 0. Sabemos como gerar tal sequência ortogonal de chips usando um método conhecido como **códigos de Walsh**. Em termos matemáticos, a ortogonalidade das sequências de chips pode ser expressa por:

$$\mathbf{S} \bullet \mathbf{T} \equiv \frac{1}{m}\sum_{i=1}^{m} S_i T_i = 0 \qquad (2.5)$$

Em linguagem comum, o número de pares iguais é igual ao de pares diferentes. Essa propriedade da ortogonalidade será essencial mais adiante. Observe que, se $\mathbf{S} \bullet \overline{\mathbf{T}} = 0$, então **S • T** também será 0. O produto interno normalizado de qualquer sequência de chips por ela mesma é igual a 1:

$$\mathbf{S} \bullet \mathbf{S} = \frac{1}{m}\sum_{i=1}^{m} S_i S_i = \frac{1}{m}\sum_{i=1}^{m} S_i^2 = \frac{1}{m}\sum_{i=1}^{m} (\pm 1)^2 = 1$$

Isso ocorre porque cada um dos *m* termos do produto interno é 1 e, portanto, a soma é *m*. Observe também que $\mathbf{S} \bullet \overline{\mathbf{S}} = -1$.

Durante cada intervalo com duração de um bit, uma estação pode transmitir um bit 1 (enviando sua sequência de chips), pode transmitir um bit 0 (enviando a parte negativa de sua sequência de chips), ou pode ficar inativa e não realizar nenhuma transmissão. Por enquanto, supomos que todas as estações estão sincronizadas e que a transmissão de todas as sequências de chips começa no mesmo instante. Quando duas ou mais estações transmitem simultaneamente, suas sequências bipolares somam-se linearmente. Por exemplo, se, durante um período de um chip, três estações transmitirem +1 e uma estação transmitir –1, o resultado +2 será recebido. Isso pode ser considerado como a soma de voltagens sobrepostas no canal: se três estações transmitirem + 1 volt e uma estação transmitir –1 volt como saída, o resultado será 2 volts. Por exemplo, a Figura 2.24(c) apresenta seis exemplos em que uma ou mais estações transmitem um bit 1 ao mesmo tempo. No primeiro exemplo, *C* transmite um bit 1 e, assim, simplesmente obtemos a sequência de chips de *C*. No segundo, *B* e *C* transmitem bits 1 e obtemos a soma de suas sequências bipolares de chips da seguinte forma:

(–1 –1 +1 –1 +1 +1 +1 –1) + (–1 +1 –1 +1 +1 +1 –1 –1)
= (–2 0 0 0 +2 +2 0 –2)

Para recuperar o fluxo de bits de uma estação individual, o receptor precisa conhecer com antecedência a sequência de chips da estação transmissora. Ele executa a recuperação calculando o produto interno normalizado da sequência de chips recebida e a sequência de chips da estação cujo fluxo de bits está tentando recuperar. Se a sequência de chips recebida for **S** e o receptor estiver tentando ouvir uma estação cuja sequência de chips é **C**, ele apenas calcula o produto interno normalizado, **S•C**.

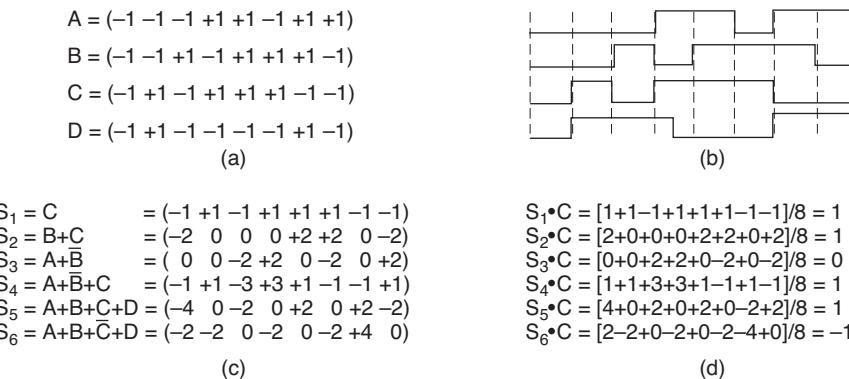



Figura 2.24 | (a) Sequências de chips binárias para quatro estações. (b) Sequências de chips bipolares. (c) Seis exemplos de transmissões. (d) Recuperação do sinal da estação *C*.

Para entender por que esse procedimento funciona, imagine que as duas estações, *A* e *C*, transmitem um bit 1 ao mesmo tempo que *B* transmite um bit 0, como no caso do terceiro exemplo. O receptor percebe a soma, $\mathbf{S} = \mathbf{A} + \overline{\mathbf{B}} + \mathbf{C}$, e calcula:

$$\mathbf{S} \bullet \mathbf{C} = (\mathbf{A} + \overline{\mathbf{B}} + \mathbf{C}) \bullet \mathbf{C} = \mathbf{A} \bullet \mathbf{C} + \overline{\mathbf{B}} \bullet \mathbf{C} + \mathbf{C} \bullet \mathbf{C} = 0 + 0 + 1 = 1$$

Os dois primeiros termos desaparecem porque todos os pares de sequências de chips foram cuidadosamente escolhidos para serem ortogonais, como mostra a Equação 2.5. Agora já deve estar claro por que essa propriedade precisa ser imposta às sequências de chips.

Para tornar mais concreto o processo de decodificação, vamos considerar novamente os seis exemplos da Figura 2.24(d). Suponha que o receptor esteja interessado em extrair o bit enviado pela estação *C* de cada uma das seis somas de S_1 a S_6. Ele calcula o bit somando aos pares os produtos da **S** recebida com o vetor **C** da Figura 2.24(a), extraindo depois 1/8 do resultado (pois, neste caso $m = 8$). Os exemplos incluem casos em que *C* é silencioso, envia um bit 1 e envia um bit 0, individualmente e em combinação com outras transmissões. Como vemos, o bit correto é decodificado a cada vez. É como falar francês.

Em princípio, dada uma capacidade de computação suficiente, o receptor poderá escutar todos os transmissores ao mesmo tempo, executando o algoritmo de decodificação correspondente a cada um deles em paralelo. Na prática, é mais fácil falar do que fazer, e é útil saber quais transmissores poderiam estar transmitindo.

O ideal é que, no sistema CDMA sem ruído que estudamos aqui, o número de estações que enviam simultaneamente possa ser arbitrariamente grande, usando sequências de chips maiores. Para 2^n estações, os códigos de Walsh podem oferecer 2^n sequências de chips ortogonais de tamanho 2^n. Contudo, uma limitação significativa é o fato de considerarmos que todos os chips são sincronizados no tempo no receptor. Essa sincronização nem mesmo é aproximadamente verdadeira em algumas aplicações, como em redes de celular (nas quais o CDMA foi bastante utilizado a partir da década de 1990). Isso leva a diferentes projetos. Voltaremos a esse assunto mais adiante no capítulo, descrevendo como o CDMA assíncrono difere do CDMA síncrono.

Assim como as redes de celular, o CDMA é usado por satélites e redes a cabo. Nesta rápida introdução, passamos brevemente pelos muitos fatores complicadores. Os engenheiros que desejam obter um conhecimento profundo do CDMA deverão ler Viterbi (1995) e Lee e Miller (1998). Entretanto, essas referências exigem muita base em engenharia de comunicação.

2.6 A rede pública de telefonia comutada

Quando dois computadores de uma mesma empresa ou organização, instalados perto um do outro, precisam se comunicar, geralmente é mais fácil conectá-los por meio de um cabo. As LANs funcionam dessa forma. No entanto, quando as distâncias começam a ficar grandes, há muitos computadores e cabos têm de atravessar uma estrada ou outra passagem pública, os custos de instalação de cabos privados costumam ser proibitivos. Além disso, em quase todos os países do mundo, também é ilegal estender linhas de transmissão privadas em (ou sob) propriedades do governo. Consequentemente, os projetistas de rede devem confiar nos recursos de telecomunicações existentes.

Esses recursos, em particular a **PSTN** (**Public Switched Telephone Network**), foram projetados há muitos anos, tendo em vista um objetivo completamente diferente: a transmissão da voz humana de uma forma mais ou menos reconhecível. Quando esses recursos são adaptados para a comunicação computador/computador, o resultado é, no máximo, tolerável. Para entender o tamanho do problema, considere que um cabo barato instalado entre dois computadores possa transferir dados a 1 Gbps ou mais. Por outro lado, a ADSL típica, alternativa bem mais rápida a

um modem de telefone, trabalha em torno de 1 Mbps. A diferença entre os dois é a diferença entre viajar em um avião e fazer uma caminhada sem nenhuma pressa.

Apesar disso, o sistema telefônico está bastante interligado com redes de computadores (longas distâncias), de modo que vale a pena dedicar algum tempo para estudá-lo em detalhes. O fator limitador para fins de rede é o 'último quilômetro' no qual os clientes se conectam, e não os troncos e switches dentro da rede telefônica. Essa situação está mudando com a implantação gradual da tecnologia de fibra e digital na borda da rede, mas ainda são necessários tempo e dinheiro. Durante a longa espera, os projetistas dos sistemas de computadores, acostumados a trabalhar com sistemas que oferecem desempenho no mínimo três ordens de grandeza maior, começam a dedicar muito tempo e esforço tentando descobrir um meio de usar a rede telefônica com maior eficiência.

Nas próximas seções, descreveremos o sistema telefônico e mostraremos como ele funciona. Para obter informações adicionais sobre o funcionamento interno do sistema telefônico, consulte Bellamy (2000).

2.6.1 Estrutura do sistema telefônico

Logo depois que Alexander Graham Bell patenteou a invenção do telefone, em 1876 (apenas algumas horas antes de seu concorrente, Elisha Gray), houve uma grande demanda por essa nova invenção. Inicialmente, o mercado estava voltado para a venda de telefones, que eram comercializados aos pares. Era o usuário quem tinha de conectar os dois aparelhos usando um fio. Se o proprietário de um telefone quisesse usar o aparelho para conversar com *n* outros proprietários de telefone, tinha de conectar fios em todas as *n* residências. Em um ano, as cidades ficaram tomadas por fios que passavam pelas casas e pelas árvores, em uma selva de fios. Logo ficou óbvio que o modelo de conexão de um telefone a outro, como é mostrado na Figura 2.25(a), não funcionaria.

Bell percebeu essa situação e criou a Bell Telephone Company, que abriu sua primeira estação de comutação (em New Haven, Connecticut) em 1878. A empresa ligava um fio até a casa ou o escritório de cada usuário. Para fazer uma chamada, o usuário girava a manivela, o que emitia um som na companhia telefônica e chamava a atenção de um operador. Este, por sua vez, conectava manualmente o emissor da chamada ao receptor usando um jumper. Observe, na Figura 2.25(b), o modelo de uma única estação de comutação.

Não demorou muito tempo para as estações de comutação da Bell System se espalharem por todos os locais, e logo as pessoas passaram a querer fazer chamadas interurbanas. Por isso, a Bell System passou a conectar uma estação de comutação a outra. Contudo, o problema original veio à tona mais uma vez: conectar cada estação de comutação a outra através de um fio entre elas logo se tornou inviável. Então, foram inventadas as estações de comutação de segundo nível. Depois de algum tempo, tornaram-se necessárias várias estações de segundo nível, como mostra a Figura 2.25(c). Mais tarde, a hierarquia cresceu até alcançar cinco níveis.

Em 1890, era possível notar a presença das três principais partes do sistema telefônico: as estações de comutação, os fios que ligavam os usuários a essas estações (agora já operando com cabos de pares trançados, isolados e balanceados em vez de cabos abertos com retorno por terra) e as conexões de longa distância existentes entre as estações de comutação. Para obter breves informações técnicas sobre o sistema telefônico e sua história, consulte Hawley (1991).

Embora tenha havido melhorias em todas as três áreas desde então, o modelo básico da Bell System continuou praticamente intacto por mais de cem anos. Embora seja bastante simplificada, a descrição a seguir apresenta a ideia básica do sistema telefônico. Cada telefone contém dois fios de cobre que saem do aparelho e se conectam diretamente à **estação final** mais próxima da companhia telefônica (também denominada **estação central local**). Em geral, a distância varia de 1 a 10 km, sendo menor nas cidades que nas regiões rurais. Só nos Estados Unidos, existem cerca 22 mil estações finais. As conexões através de dois fios entre o telefone de cada assinante e a estação final são conhecidas no mercado como **circuito terminal**. Se todos os circuitos terminais existentes no mundo inteiro fossem esticados de

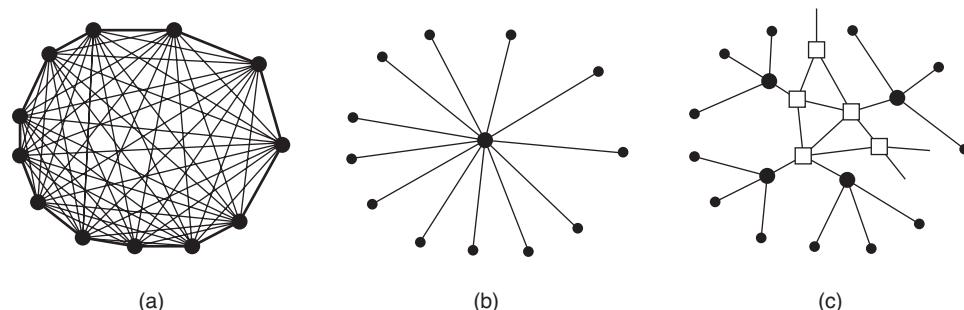



Figura 2.25 (a) Rede totalmente interconectada. (b) Switch centralizado. (c) Hierarquia de dois níveis.

uma extremidade a outra, seu comprimento equivaleria a mil vezes a distância da Terra à Lua e de volta à Terra.

Houve uma época em que 80 por cento do capital da AT&T estava no cobre dos circuitos terminais, o que a tornava a maior mina de cobre do mundo. Felizmente, essa informação não foi muito difundida na comunidade financeira. Se tivessem conhecimento desse fato, alguns empresários poderiam comprar a AT&T, encerrar todos os serviços de telefonia dos Estados Unidos, descascar toda a fiação e vender os fios a uma refinaria de cobre para ter um retorno rápido do capital.

Se um assinante conectado a uma determinada estação final ligar para outro assinante da mesma estação, o mecanismo de comutação dentro da estação configurará uma conexão elétrica direta entre os dois circuitos terminais. Essa conexão permanecerá intacta durante a chamada.

Se o telefone chamado estiver conectado a outra estação final, outro procedimento terá de ser usado. Cada estação final contém uma série de linhas de saída para um ou mais centros de comutação vizinhos, denominados **estações interurbanas** (ou, se estiverem na mesma área, **estações Tandem**). Essas linhas são denominadas **troncos de conexão interurbana**. O número de diferentes tipos de centros de comutação e sua topologia variam de um país para outro, dependendo da densidade da telefonia do país.

Se as estações finais do transmissor e do receptor tiverem um tronco de conexão interurbana ligado à mesma estação interurbana (uma situação bastante provável caso estejam geograficamente próximos), a conexão poderá ser estabelecida dentro da estação interurbana. Observe, na Figura 2.25(c), uma rede telefônica formada apenas por telefones (os pontos pequenos), estações finais (os pontos maiores) e estações interurbanas (os quadrados).

Se o transmissor e o receptor não compartilharem a mesma estação interurbana, terá de ser estabelecido um caminho entre duas estações interurbanas. As estações interurbanas se comunicam entre si por meio de **troncos interurbanos** de alta largura de banda (também denominados **troncos entre estações**). Antes da dissolução da AT&T, em 1984, o sistema telefônico dos Estados Unidos usava o roteamento hierárquico para encontrar um caminho, indo para níveis mais altos da hierarquia até que houvesse uma estação de comutação em comum. Isso foi substituído pelo roteamento não hierárquico, mais flexível. A Figura 2.26 mostra como uma conexão de média distância pode ser roteada.

Nas telecomunicações, são usados vários meios de transmissão. Diferente dos prédios de escritórios modernos, nos quais a fiação normalmente é de Categoria 5, os circuitos terminais são formados por cabos de pares trançados de Categoria 3, com a fibra apenas começando a aparecer. Entre as estações de comutação, o uso de cabos coaxiais, micro-ondas e, principalmente, de fibras ópticas, é bastante frequente.

No passado, a transmissão em todo o sistema telefônico era analógica, com o sinal de voz real sendo transmitido como uma corrente elétrica da origem até o destino. Com o advento da fibra óptica, da eletrônica digital e dos computadores, todos os troncos e switches são agora digitais, deixando o circuito terminal como o último fragmento de tecnologia analógica no sistema. A transmissão digital é preferida porque, em uma chamada longa, não é necessário reproduzir com precisão uma forma de onda analógica depois de ter passado por muitos amplificadores. Ser capaz de distinguir corretamente 0 de 1 já é suficiente. Essa propriedade torna a transmissão digital mais confiável que a analógica. Ela também é mais econômica e de manutenção mais fácil.

Em suma, o sistema telefônico é formado por três componentes principais:

1. Circuitos terminais (pares trançados analógicos indo para as residências e para as empresas).
2. Troncos (fibra óptica digital conectando as estações de comutação).
3. Estações de comutação (onde as chamadas são transferidas de um tronco para outro).

Depois de uma rápida análise da política das companhias telefônicas, voltaremos a cada um desses três componentes e os analisaremos em detalhes. Os circuitos terminais oferecem acesso ao sistema inteiro para todas as

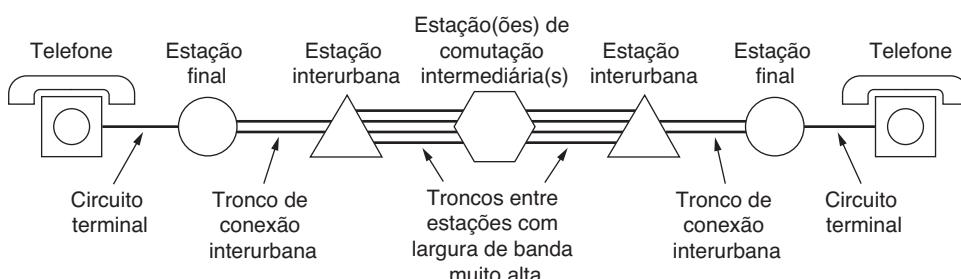



Figura 2.26 | Rota de um circuito típico para uma chamada de longa distância.

pessoas; assim, eles são críticos. Infelizmente, eles também constituem o elo mais fraco no sistema. Para os troncos de longa distância, a principal questão é reunir várias chamadas e transmiti-las ao mesmo tempo, pela mesma fibra. Esse assunto é chamado multiplexação, e para realizar esse processo aplicamos FDM e TDM. Por último, existem duas formas fundamentalmente distintas de executar a comutação; portanto, analisaremos ambas.

2.6.2 A política das companhias telefônicas

Por muitas décadas até 1984, a Bell System foi a responsável pelo serviço de chamadas locais e interurbanas nos Estados Unidos. Na década de 1970, o governo norte-americano concluiu que esse era um monopólio ilegal e promoveu uma ação para desmembrá-lo. O governo foi vitorioso e, em 1º de janeiro de 1984, a AT&T foi dividida na AT&T Long Lines, em 23 **BOCs** (**Bell Operating Companies**) e em algumas outras partes. As 23 BOCs foram agrupadas em sete BOCs regionais (RBOCs), o que as tornou economicamente viáveis. Toda a natureza do sistema de telecomunicações norte-americano foi alterada da noite para o dia por uma ordem judicial (e *não* por um ato do Congresso norte-americano).

Os detalhes exatos dessa ruptura foram descritos no conhecido **julgamento final modificado,** ou **MFJ** (**Modified Final Judgement**), um oxímoro, se é que houve um — se o julgamento podia ser modificado, isso significa que ele não era o resultado final. Esse fato provocou o aumento da concorrência, a melhoria dos serviços e a redução dos preços para consumidores e empresas. Entretanto, os preços para o serviço local cresceram à medida que os subsídios cruzados das chamadas de longa distância foram eliminados, e o serviço local teve de se tornar autossuficiente. Hoje em dia, muitos outros países estão considerando a abertura à concorrência em termos semelhantes.

De relevância direta para nossos estudos é que a nova estrutura competitiva introduziu um recurso técnico fundamental para a arquitetura da rede telefônica. Para determinar precisamente a quem cabiam as responsabilidades, o território dos Estados Unidos foi dividido em 164 **áreas de transporte e acesso local,** ou **LATAs** (**Local Access and Transport Areas**). De uma forma bem genérica, uma LATA corresponde à região coberta por um único código de área. Em geral, dentro de uma LATA existia um **portador local de troca,** ou **LEC** (**Local Exchange Carrier**) que detinha o monopólio do sistema telefônico convencional dentro de sua área. Os LECs mais importantes eram as BOCs, embora algumas LATAs contivessem uma ou mais das 1.500 companhias telefônicas independentes que operavam como LECs.

A nova característica era que todo o tráfego entre LATAs passou a ser manipulado por um tipo diferente de empresa, um **portador Interexchange**, ou **IXC** (**IntereXchange Carrier**). Originalmente, a AT&T Long Lines era o único IXC seguro, mas hoje em dia existem concorrentes fortes, como Verizon e Sprint, no ramo dos IXCs. Uma das preocupações ao ocorrer o desmembramento foi assegurar-se que todos os IXCs seriam tratados igualmente em termos de qualidade das linhas, das tarifas e do número de dígitos que seus clientes teriam de teclar para usá-los. Observe, na Figura 2.27, como essa situação é tratada. Nessa figura, vemos três exemplos de LATAs, cada uma com várias estações finais. As LATAs 2 e 3 também têm uma

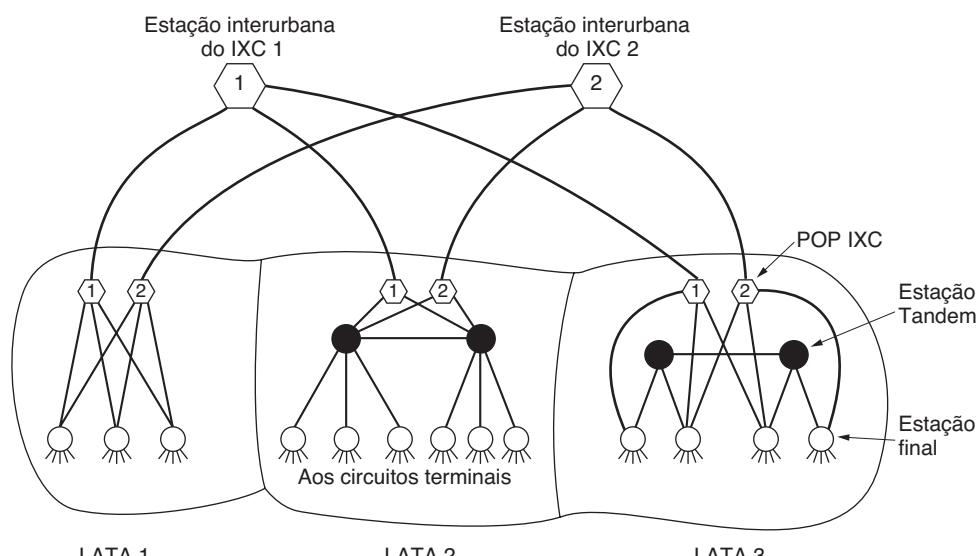



Figura 2.27 | O relacionamento entre LATAs, LECs e IXCs. Todos os círculos são estações de comutação de LECs. Cada hexágono pertence ao IXC indicado pelo número.

pequena hierarquia com estações Tandem (estações interurbanas intraLATA).

Qualquer IXC que deseje se encarregar de chamadas provenientes de uma LATA pode criar uma estação de comutação denominada **ponto de presença**, ou **POP** (**Point of Presence**). O LEC é necessário para conectar cada IXC a cada estação final, seja diretamente, como nas LATAs 1 e 3, seja indiretamente, como na LATA 2. Além disso, as condições da conexão, tanto técnicas quanto financeiras, têm de ser idênticas para todos os IXCs. Dessa forma, um assinante da LATA 1, por exemplo, pode escolher qual IXC usará para entrar em contato com assinantes que façam parte da LATA 3.

Como parte do MFJ, os IXCs foram proibidos de prestar serviços telefônicos locais, e os LECs foram proibidos de prestar serviços telefônicos entre LATAs, apesar de todas serem livres para atuar em quaisquer outros ramos, como pequenos restaurantes. Em 1984, essa era uma condição razoavelmente não ambígua. Infelizmente, a tecnologia tem uma forma interessante de tornar a lei obsoleta. Nem a TV a cabo nem os telefones celulares foram cobertos pelo acordo. À medida que a TV a cabo passou de unidirecional para bidirecional e a popularidade dos telefones celulares explodiu, os LECs e os IXCs começaram a comprar ou a se associar às operadoras de TV a cabo ou de telefones celulares.

Em 1995, o Congresso dos Estados Unidos percebeu que tentar manter uma distinção entre os vários tipos de empresas não era mais sustentável e elaborou um projeto de lei que permitiria às empresas de TV a cabo, companhias telefônicas locais, concessionárias de comunicação de longa distância e operadoras de sistemas celulares entrarem nos ramos de negócio umas das outras. A ideia era que qualquer empresa poderia oferecer a seus clientes um único pacote integrado contendo serviços de TV a cabo, telefone e informações, e que diferentes empresas seriam concorrentes em serviços e preços. O projeto de lei foi sancionado em fevereiro de 1996, como uma reestruturação importante da regulamentação das telecomunicações. Como resultado, algumas BOCs se tornaram IXCs e algumas outras empresas, como operadoras de TV a cabo, começaram a oferecer serviços de telefonia local, competindo com os LECs.

Uma propriedade interessante da lei de 1996 é a exigência de que os LECs implementem **portabilidade de número local**. Isso significa que um cliente pode mudar de companhia telefônica local sem ter de receber um novo número de telefone. A portabilidade para números de telefone móvel (e entre linhas fixas e móveis) seguiu o exemplo em 2003. Essas providências removeram um enorme obstáculo para muitas pessoas, tornando-as muito mais inclinadas a mudar de LECs. Como resultado, o cenário das telecomunicações dos Estados Unidos se tornou muito mais competitivo, e outros países estão seguindo o mesmo caminho. Com frequência, outros países esperam para ver como esse tipo de experiência funciona nos Estados Unidos. Se dá certo, eles fazem o mesmo; se funciona mal, eles tentam algo diferente.

2.6.3 O circuito terminal: modems, ADSL e fibra óptica

Agora chegou o momento de iniciarmos nosso estudo detalhado do funcionamento do sistema de telefonia. Vamos começar pela parte com a qual a maioria das pessoas está familiarizada: o circuito terminal de dois fios, que vem da estação final de uma companhia telefônica até residências e pequenas empresas. Com frequência, o circuito terminal também é chamado 'o último quilômetro', embora o comprimento real possa chegar a vários quilômetros. Ele utiliza sinalização analógica há mais de cem anos e é provável que continue a utilizá-la por mais alguns anos, em virtude do custo elevado da conversão para sinalização digital.

Muito esforço tem sido dedicado para espremer a rede de dados pelos circuitos terminais de cobre que já estão instalados. Os modems de telefone enviam dados digitais entre computadores pelo canal estreito que a rede telefônica oferece para uma chamada de voz. Eles já foram muito utilizados, mas foram substituídos em grande parte por tecnologias de banda larga, como ADSL, que reutilizam o circuito terminal para enviar dados digitais de um cliente para a estação final, onde são enviados para a Internet. Os modems e a ADSL precisam lidar com as limitações dos antigos circuitos terminais: largura de banda relativamente estreita, atenuação e distorção dos sinais, e susceptibilidade ao ruído elétrico, como linha cruzada.

Em alguns lugares, o circuito terminal foi modernizado instalando-se fibra óptica até a residência (ou muito próximo dela). A fibra é o caminho do futuro. Essas instalações admitem redes de computadores desde o início, com o circuito terminal tendo ampla largura de banda para os serviços de dados. O fator limitante é o que as pessoas pagarão, e não a física do circuito terminal.

Nesta seção, estudaremos o circuito terminal, tanto o antigo quanto o novo. Veremos os modems de telefone, ADSL e *Fiber to the Home*.

Modems de telefone

Para enviar bits pelo circuito terminal, ou qualquer outro canal físico pelo mesmo motivo, eles precisam ser convertidos para sinais analógicos que podem ser transmitidos pelo canal. Essa conversão é realizada usando os métodos para modulação digital que estudamos na seção anterior. Na outra ponta do canal, o sinal analógico é convertido de volta para bits.

Um dispositivo que converte entre um fluxo de bits digitais e um sinal analógico que representa os bits é chamado **modem**, que é uma abreviação de '*mo*dulador-*de*-*mo*dulador'. Os modems vêm em muitas variedades: modems de te-

lefone, DSL, a cabo, sem fio etc. O modem pode ser embutido no computador (o que agora é comum para modems de telefone) ou uma caixa separada (comum para modems DSL e a cabo). Logicamente, o modem é inserido entre o computador (digital) e o sistema telefônico (analógico), como vemos na Figura 2.28.

Os modems de telefone são usados para enviar bits entre dois computadores por uma linha telefônica com qualidade de voz, em lugar da conversação que normalmente preenche a linha. A dificuldade principal ao fazer isso é que uma linha de telefone para a voz humana é limitada a 3.100 Hz, suficiente para transportar uma conversa. Essa largura de banda é mais de quatro ordens de grandeza menor que a largura de banda usada para Ethernet ou rede 802.11 (WiFi). Não é surpresa que as taxas de dados dos modems de telefone também sejam quatro ordens de grandeza menores que as da Ethernet e da rede 802.11.

Vamos analisar os números para ver por que isso acontece. O teorema de Nyquist nos diz que, mesmo com uma linha perfeita de 3.000 Hz (que uma linha telefônica decididamente não é), não há sentido em enviar símbolos em uma taxa mais rápida que 6.000 bauds. Na prática, a maioria dos modems envia a uma taxa de 2.400 símbolos/s, ou 2.400 bauds, e detecta e captura vários bits por símbolo, enquanto permite o tráfego nas duas direções ao mesmo tempo (usando diferentes frequências para diferentes direções).

O humilde modem de 2.400 bps usa 0 volt para um valor lógico 0 e 1 volt para um valor 1, com 1 bit por símbolo. No próximo passo, ele pode usar quatro símbolos diferentes, como nas quatro fases do QPSK, de modo que, com 2 bits/símbolo, ele pode alcançar uma taxa de dados de 4.800 bps.

Uma longa progressão de taxas mais altas tem sido alcançada enquanto a tecnologia melhora. Taxas mais altas exigem um conjunto maior de símbolos, ou **diagrama de constelação**. Com muitos símbolos, até mesmo uma pequena quantidade de ruído na amplitude ou fase detectada pode resultar em um erro. Para reduzir a chance de erros, os padrões para modems de velocidade mais alta utilizam alguns dos símbolos para correção de erros. Os esquemas são conhecidos como **modulação codificada por treliças**, ou **TCM** (**Trellis Coded Modulation**) (Ungerboeck, 1987).

O modem **V.32** utiliza 32 pontos do diagrama de constelação para transmitir 4 bits de dados e 1 bit de paridade por símbolo a 2.400 bauds, a fim de alcançar 9.600 bps com correção de erros. O próximo passo acima de 9.600 bps é 14.400 bps. Ele é chamado **V.32 bis**, e transmite 6 bits de dados e 1 bit de paridade por amostra a 2.400 bauds. O modem de telefone que segue o V.32 bis é o **V.34**, que funciona em 28.800 bps a 2.400 bauds com 12 bits de dados/símbolo. O último modem dessa série é o **V.34 bis**, que utiliza 14 bits de dados/símbolo a 2.400 bauds para atingir 33.600 bps.

Por que parar aqui? A razão pela qual os modems-padrão param em 33.600 é que o limite de Shannon para o sistema telefônico é de aproximadamente 35 kbps, com base no comprimento médio dos circuitos terminais e na qualidade dessas linhas. Uma transmissão mais rápida que isso violaria as leis da física (departamento da termodinâmica).

Contudo, existe um meio de mudar a situação. Na estação final da companhia telefônica, os dados são convertidos para a forma digital para transmissão dentro da rede telefônica (o núcleo da rede telefônica convertia de analógico para digital há muito tempo). O limite de 35 kbps é para a situação em que existem dois circuitos terminais, um em cada ponta. Cada um deles acrescenta ruído ao sinal. Se pudéssemos nos livrar de um desses circuitos terminais, aumentaríamos a SNR e a taxa máxima seria dobrada.

É com essa técnica que os modems de 56 kbps funcionam. Uma extremidade, normalmente um ISP, recebe uma entrada digital de alta qualidade da estação final mais próxima. Desse modo, quando uma extremidade da conexão é um sinal de alta qualidade, como ocorre com a maioria dos ISPs atuais, a taxa máxima de dados pode chegar a 70 kbps. Entre dois usuários domésticos com modems e linhas analógicas, o máximo ainda é 33,6 kbps.

A razão para o uso de modems de 56 kbps (em vez de modems de 70 kbps) está relacionada ao teorema de Nyquist. Um canal telefônico é usado dentro do sistema telefônico como amostras digitais. Cada canal telefônico tem

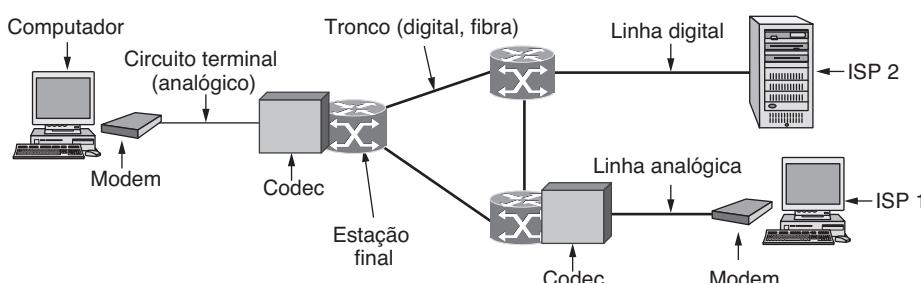



Figura 2.28 | O uso de transmissão analógica e digital para uma chamada de computador para computador. A conversão é feita por modems e codecs.

cerca de 4.000 Hz de largura, incluindo as bandas de proteção. O número máximo de amostras independentes por segundo é, portanto, 8.000. O número de bits por amostra nos Estados Unidos é 8, um dos quais pode ser usado para fins de controle, permitindo 56.000 bits/s de dados do usuário. Na Europa, todos os 8 bits estão disponíveis para os usuários, e assim poderiam ser usados modems de 64.000 bits/s; porém, para chegar a um acordo internacional sobre um padrão, o valor de 56.000 foi escolhido.

O resultado final são os modems-padrão **V.90** e **V.92**. Eles são capazes de transmitir 56 kbps no canal downstream (ISP ao usuário) e 33,6 kbps e 48 kbps no canal upstream (usuário para ISP), respectivamente. A assimetria ocorre porque normalmente existem mais dados transportados do ISP para o usuário do que o contrário. Isso também significa que uma parte maior da largura de banda limitada pode ser alocada ao canal downstream, para aumentar as chances de que ele realmente funcione a 56 kbps.

Linhas digitais do assinante

Quando a indústria de telefonia finalmente conseguiu alcançar 56 kbps, ela se congratulou pelo serviço benfeito. Enquanto isso, a indústria de TV a cabo estava oferecendo velocidades de até 10 Mbps sobre cabos compartilhados. À medida que o acesso à Internet se tornou uma parte cada vez mais importante de seus negócios, as companhias telefônicas (LECs) começaram a perceber que precisavam de um produto mais competitivo. Sua resposta foi começar a oferecer novos serviços digitais ao circuito terminal do assinante.

Inicialmente, havia muitas ofertas sobrepostas, todas sob o nome genérico de **linha digital do assinante**, ou **xDSL** (**Digital Subscriber Line**), para diversos *x*. Os serviços com mais largura de banda do que o serviço telefônico padrão às vezes são chamados de **banda larga** (**broadband**), embora o termo seja mais um conceito de marketing do que um conceito técnico específico. Mais adiante, vamos nos concentrar principalmente naquele que provavelmente se tornou o mais popular desses serviços, a **ADSL** (**Asymmetric DSL**). Também usaremos o termo DSL ou xDSL como uma abreviação geral.

A razão para os modems serem tão lentos é que os telefones foram inventados para transportar a voz humana, e o sistema inteiro foi cuidadosamente otimizado para esse propósito. Os dados sempre estiveram em segundo plano. No ponto em que cada circuito terminal chega à estação final, o fio passa por um filtro que atenua todas as frequências abaixo de 300 Hz e acima de 3.400 Hz. O corte não é nítido — 300 Hz e 3.400 Hz são os pontos de 3 dB — assim, a largura de banda normalmente é mencionada como 4.000 Hz, embora a distância entre os pontos de 3 dB seja de 3.100 Hz. Portanto, os dados também estão restritos a essa banda estreita.

O artifício que faz o xDSL funcionar é o fato de que, quando um cliente se inscreve nele, a linha de entrada é conectada a um tipo diferente de switch, que não tem esse filtro, assim tornando disponível toda a capacidade do circuito terminal. Então, o fator limitador passa a ser a constituição física do circuito terminal, não a largura de banda artificial de 3.100 Hz criada pelo filtro.

Infelizmente, a capacidade do circuito terminal cai rapidamente com a distância da estação final, pois o sinal é degradado cada vez mais ao longo do fio. Ela também depende da espessura e da qualidade geral do par trançado. A Figura 2.29 mostra um esboço da largura de banda potencial como uma função da distância. Essa figura pressupõe que todos os outros fatores estão otimizados (novos fios, pacotes de serviços etc.).

A implicação dessa figura cria um problema para a companhia telefônica. Quando escolhe uma velocidade para oferecer, ela está ao mesmo tempo escolhendo um raio a partir de suas estações finais, além do qual o serviço não poderá ser oferecido. Isso significa que, quando clientes distantes tentarem assinar o serviço, eles receberão a seguinte mensagem: 'Muito obrigado por seu interesse, mas você está cem metros além da distância máxima da central mais próxima que poderia lhe oferecer o serviço.

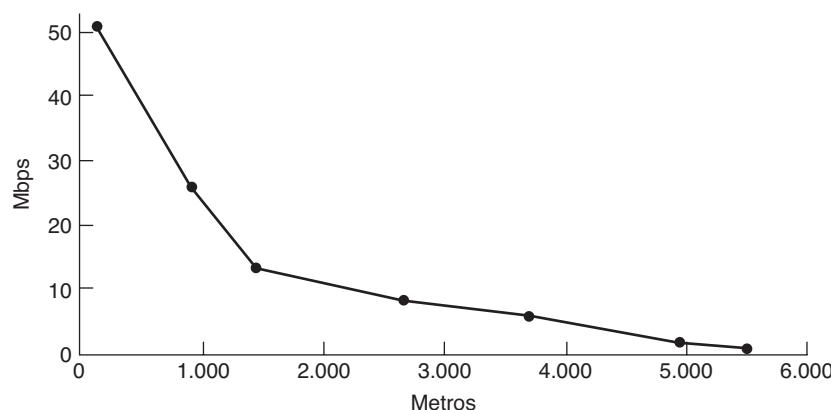



Figura 2.29 | Variação da largura de banda *versus* a distância sobre o UTP da categoria 3 para DSL.

Você não gostaria de se mudar?' Quanto mais baixa a velocidade escolhida, maior o raio e maior o número de clientes cobertos. Porém, quanto mais baixa a velocidade, menos atraente será o serviço e menor o número de pessoas que estarão dispostas a pagar por ele. É aqui que os negócios encontram a tecnologia.

Todos os serviços xDSL foram criados visando a certos objetivos. Primeiro, os serviços devem funcionar nos circuitos terminais de pares trançados da Categoria 3 existente. Segundo, eles não devem afetar os telefones e os aparelhos de fax atuais dos clientes. Terceiro, eles devem ser muito mais rápidos que 56 kbps. Quarto, eles devem estar sempre ativos, apenas com uma tarifa mensal e nenhuma tarifa por minuto.

Para cumprir os objetivos técnicos, o espectro de 1,1 MHz disponível no circuito terminal é dividido em 256 canais independentes de 4.312,5 Hz cada. Esse arranjo pode ser visto na Figura 2.30. O esquema OFDM, que vimos na seção anterior, é usado para enviar dados por esses canais, embora normalmente seja chamado **DMT** (**Discrete MultiTone**) no contexto da ADSL. O canal 0 é usado para o **POTS** (**Plain Old Telephone Service**). Os canais de 1 a 5 não são usados, a fim de impedir que o sinal de voz e os sinais de dados interfiram uns com os outros. Dos 250 canais restantes, um é utilizado para o controle upstream e outro é empregado para o controle downstream. Os outros canais estão disponíveis para dados do usuário.

Em princípio, cada um dos canais restantes pode ser usado em um fluxo de dados full-duplex; porém, harmônicos, linhas cruzadas e outros efeitos mantêm a utilização de sistemas práticos bem abaixo do limite teórico. Cabe ao provedor definir quantos canais serão usados para upstream e quantos para downstream. Uma mistura de 50 por cento para cada um é tecnicamente possível, mas a maioria dos provedores aloca algo como 80 por cento a 90 por cento da largura de banda ao canal downstream, pois a maioria dos usuários faz mais download do que upload de dados. Essa escolha deu origem à letra 'A' no acrônimo ADSL (assimétrica). Uma divisão comum reserva 32 canais para upstream e os restantes para downstream. Também é possível tornar bidirecionais alguns dos canais upstream mais altos para aumentar a largura de banda, embora essa otimização exija o uso de um circuito especial para cancelamento de ecos.

O padrão ADSL internacional, conhecido como **G.dmt**, foi aprovado em 1999. Ele permite velocidades de até 8 Mbps downstream e 1 Mbps upstream. Ele foi substituído por uma segunda geração em 2002, chamada ADSL2, com diversas melhorias para permitir velocidades de até 12 Mbps downstream e 1 Mbps upstream. Agora, temos a ADSL2+, que dobra a velocidade downstream para 24 Mbps ao dobrar a largura de banda para usar 2,2 MHz pelo par trançado.

Porém, os números indicados aqui são velocidades otimistas para linhas boas e próximas (dentro de 1 a 2 km) da estação. Poucas linhas admitem essas velocidades, e poucos provedores as oferecem. Normalmente, os provedores oferecem algo como 1 Mbps downstream e 256 kbps upstream (serviço padrão), 4 Mbps downstream e 1 Mbps upstream (serviço melhorado) e 8 Mbps downstream e 2 Mbps upstream (serviço premium).

Dentro de cada canal, a modulação QAM é usada a uma taxa de aproximadamente 4.000 símbolos/s. A qualidade da linha em cada canal é monitorada constantemente, e a taxa de dados é ajustada usando um diagrama de constelação maior ou menor, como os da Figura 2.19. Diferentes canais podem ter diferentes taxas de dados, com até 15 bits por símbolo enviado em um canal com uma SNR alta, e descendo até 2, 1 ou nenhum bit por símbolo enviado em um canal com uma SNR baixa, dependendo do padrão.

A Figura 2.31 mostra uma organização ADSL típica. Nesse esquema, um técnico da companhia telefônica deve instalar um **dispositivo de interface de rede**, ou **NID** (**Network Interface Device**) no local do cliente. Essa pequena caixa plástica marca o fim da propriedade da companhia telefônica e o início da propriedade do cliente. Próximo ao NID (ou às vezes combinado a ele) há um **divisor**, um filtro analógico que separa a banda de 0 a 4.000 Hz utilizada pelo POTS dos dados. O sinal do POTS é roteado até o telefone ou o equipamento de fax existente, e o sinal de dados é roteado até um modem ADSL, que usa o processamento do sinal digital para implementar OFDM. Como a maioria dos modems ADSL atuais é externa, o computador deve estar conectado ao modem em alta velocidade. Normalmente, isso é feito usando Ethernet, um cabo USB ou rede 802.11.

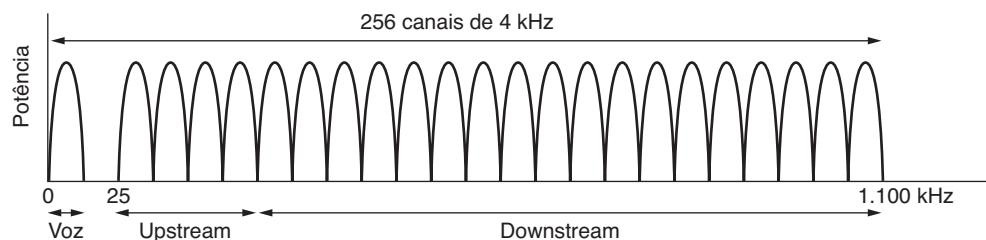



Figura 2.30 | Operação ADSL usando modelagem discreta por multitom.

Na outra extremidade do fio, no lado da estação final, está instalado um divisor correspondente. Aqui, a porção de voz do sinal é filtrada e enviada ao switch de voz normal. O sinal acima de 26 kHz é roteado para um novo tipo de dispositivo, chamado **multiplexador de acesso à linha digital do assinante**, ou **DSLAM** (**Digital Subscriber Line Access Multiplexer**), que contém a mesma espécie de processador de sinal digital que o modem ADSL. Uma vez que o sinal digital é recuperado em um fluxo de bits, são formados pacotes que são enviados ao ISP.

Essa separação completa entre o sistema de voz e a ADSL torna relativamente fácil para uma companhia telefônica distribuir esse serviço. Basta adquirir um DSLAM e um divisor, e conectar os assinantes do ADSL ao divisor. Outros serviços de alta largura de banda (por exemplo, ISDN) exigem mudanças muito maiores no equipamento de comutação existente.

Uma desvantagem do projeto da Figura 2.31 é a necessidade do NID e do divisor no local do cliente. A instalação desses itens só pode ser feita por um técnico da companhia telefônica, necessitando de um dispendioso serviço de assistência (isto é, enviar um técnico até o local do cliente). Portanto, também teve de ser padronizado um projeto alternativo sem divisores, informalmente chamado **G.lite**. Ele é idêntico ao da Figura 2.31, mas sem o divisor. A linha telefônica existente é usada como está. A única diferença é a inserção de um microfiltro em cada tomada de telefone, entre o aparelho de telefone ou modem ADSL e o fio. O microfiltro para o telefone é um filtro passa-baixa, que elimina frequências acima de 3.400 Hz; o microfiltro para o modem ADSL é um filtro passa-alta, que elimina frequências abaixo de 26 kHz. Porém, esse sistema não é tão confiável quanto um divisor, e assim o G.lite só pode ser usado até 1,5 Mbps (contra 8 Mbps para a ADSL com um divisor). Para obter mais informações sobre ADSL, consulte Starr (2003).

Fiber To The Home

Os circuitos terminais de cobre instalados limitam o desempenho da ADSL e dos modems de telefone. Para que eles ofereçam serviços mais rápidos e melhores, as companhias telefônicas estão atualizando os circuitos terminais a cada oportunidade, instalando fibra óptica até casas e escritórios. O resultado é chamado **FTTH** (**Fiber To The Home**). Embora a tecnologia FTTH tenha estado disponível já um bom tempo, as instalações só começaram a ser feitas em massa em 2005, com o crescimento da demanda por Internet de alta velocidade dos clientes acostumados com DSL e cabo que queriam baixar filmes. Por volta de 4 por cento das casas nos Estados Unidos agora estão conectadas à FTTH com velocidades de acesso à Internet de até 100 Mbps.

Existem diversas variações na forma 'FTTX' (onde *X* significa porão, calçada ou vizinhança). Elas são usadas para indicar que a implantação da fibra pode chegar perto da casa. Nesse caso, o cobre (par trançado ou cabo coaxial) oferece velocidades rápidas o suficiente pela última distância curta. A escolha da extensão em que a fibra é disposta é uma questão econômica, balanceando custo com receita esperada. De qualquer forma, o importante é que a fibra óptica atravessou a barreira tradicional do 'último quilômetro'. Focalizaremos o FTTH em nossa discussão.

Assim como os fios de cobre antes dele, o circuito terminal de fibra é passivo. Isso significa que nenhum equipamento energizado é necessário para amplificar ou processar os sinais de alguma outra forma. A fibra simplesmente transporta sinais entre a casa e a estação final. Isso, por sua vez, reduz o custo e melhora a confiabilidade.

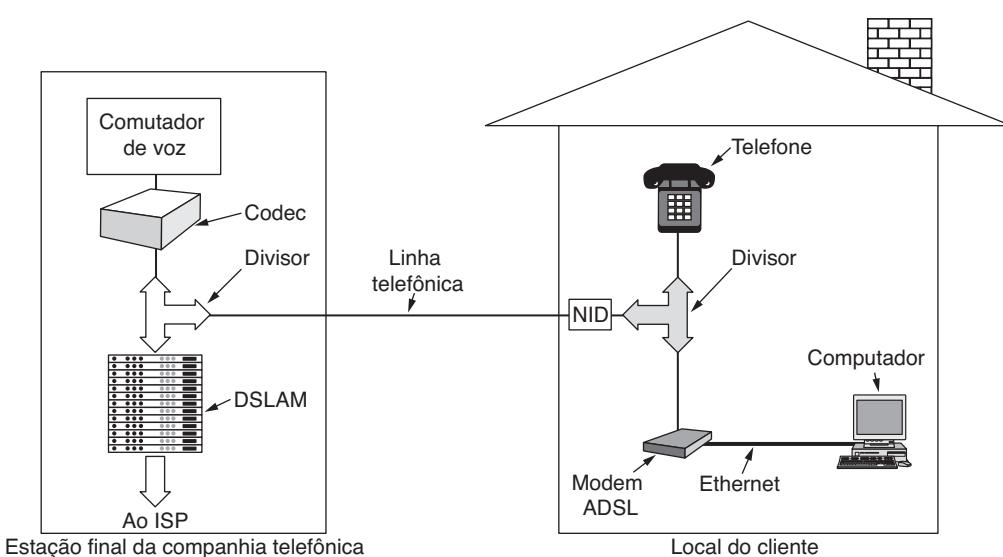



Figura 2.31 | Uma configuração típica de equipamento ADSL.

Normalmente, as fibras das casas são reunidas de modo que apenas uma fibra alcance a estação final por grupo de até cem casas. Na direção downstream, os divisores ópticos dividem o sinal da estação final, de modo que alcance todas as casas. A criptografia é necessária por segurança se apenas uma casa puder ser capaz de decodificar o sinal. Na direção upstream, colimadores ópticos mesclam os sinais das casas para um único sinal, que é recebido na estação final.

Essa arquitetura é chamada **rede óptica passiva**, ou **PON** (**Passive Optical Network**), mostrada na Figura 2.32. É comum usar um comprimento de onda compartilhado entre todas as casas para a transmissão downstream, e outro comprimento de onda para a transmissão upstream.

Mesmo com a divisão, uma enorme largura de banda e a baixa atenuação da fibra significam que as PONs podem oferecer altas velocidades aos usuários por distâncias de até 20 km. As taxas de dados reais e outros detalhes dependem do tipo de PON. Dois tipos são comuns. Como as **GPONs** (**Gigabit-capable PONs**) vêm do mundo das telecomunicações, elas são definidas por um padrão ITU. As **EPONs (Ethernet PONs)** estão mais ligadas ao mundo das redes, de modo que são definidas por um padrão IEEE. Ambas trabalham em torno de um gigabit e podem transportar tráfego para diferentes serviços, incluindo Internet, vídeo e voz. Por exemplo, as GPONs oferecem 2,4 Gbps downstream e 1,2 ou 2,4 Gbps upstream.

Para compartilhar a capacidade de uma única fibra na estação final entre diferentes casas, é preciso que haja algum protocolo. A direção downstream é fácil. A estação final pode enviar mensagens a cada casa diferente na ordem que desejar. Na direção upstream, porém, as mensagens de diferentes casas não podem ser enviadas ao mesmo tempo, ou diferentes sinais colidiriam. As casas também não podem escutar as transmissões umas das outras, de modo que não podem escutar antes de transmitir. A solução é que o equipamento nas casas solicite e receba fatias de tempo para utilizar o equipamento na estação final. Para que isso funcione, existe um processo de localização para ajustar os tempos de transmissão a partir das casas, de modo que todos os sinais recebidos na estação final sejam sincronizados. O projeto é semelhante aos modems a cabo, que explicaremos mais adiante neste capítulo. Para obter mais informações sobre o futuro das PONs, veja Grobe e Elbers (2008).

2.6.4 Troncos e multiplexação

Na rede telefônica, os troncos não são apenas muito mais rápidos do que os circuitos terminais; eles são diferentes em dois outros aspectos. O núcleo da rede telefônica transporta informações digitais, não informações analógicas, ou seja, bits e não voz. Isso necessita de uma conversão na estação final para a forma digital, para a transmissão por troncos de longa distância. Os troncos transportam milhares, ou mesmo milhões, de chamadas simultaneamente. Esse compartilhamento é importante para conseguir economias de escala, pois custa basicamente a mesma coisa instalar e manter um tronco de alta largura de banda e um de baixa largura de banda entre duas estações de comutação. Isso é realizado com as versões de multiplexação TDM e FDM.

A seguir, examinaremos rapidamente como os sinais de voz são digitalizados de modo que possam ser transportados pela rede telefônica. Depois disso, veremos como o TDM é usado para transportar bits nos troncos, incluindo o sistema TDM usado para fibra óptica (SONET). Depois, passaremos para WDM conforme se aplica à fibra óptica, que é chamada multiplexação por divisão de comprimento de onda.

Sinais de voz digitalizada

Desde cedo no desenvolvimento da rede telefônica, o núcleo lidava com chamadas de voz como informação analógica. As técnicas de FDM foram usadas por muitos anos para multiplexar canais de voz de 4.000 Hz (compostos de 3.100 Hz mais bandas de proteção) em unidades cada vez maiores. Por exemplo, 12 chamadas na banda de 60 a 108 kHz são conhecidas como um **grupo**, cinco grupos (um total de 60 chamadas) são conhecidos como um **supergrupo** e assim por diante. Esses métodos de FDM ainda são usados em alguns canais de fio de cobre e micro-ondas. Contudo, a FDM requer circuitos analógicos e isso não é adequado para um computador digital. Ao contrário, a TDM pode ser tratada inteiramente pela eletrônica digital, de modo que se tornou muito mais usada nos últimos anos.

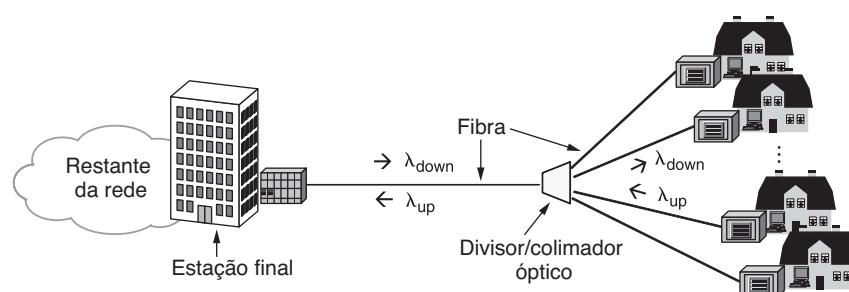



Figura 2.32 | Rede óptica passiva para Fiber To The Home.

Como a TDM só pode ser usada para dados digitais e os circuitos terminais produzem sinais analógicos, é preciso que haja uma conversão de analógico para digital na estação final, onde todos os circuitos terminais individuais se juntam para serem combinados nos troncos de saída.

Os sinais analógicos são digitalizados na estação final por um dispositivo chamado **codec** (abreviação de '*co*dificador-*dec*odificador'). O codec cria 8.000 amostras por segundo (125 μs/amostra), pois o teorema de Nyquist diz que isso é suficiente para captar todas as informações de largura de banda do canal telefônico de 4 kHz. A uma taxa de amostragem mais baixa, as informações se perderiam; a uma taxa mais alta, nenhuma informação extra seria obtida. Cada amostra da amplitude do sinal é quantizada para um número de 8 bits.

Essa técnica é chamada **modulação por código de pulso**, ou **PCM** (**Pulse Code Modulation**). Ela forma o núcleo do sistema telefônico moderno. Como consequência, praticamente todos os intervalos de tempo no sistema telefônico são múltiplos de 125 μs. A taxa de dados não compactada padrão para uma chamada telefônica com qualidade de voz é, portanto, de 8 bits a cada 125 μs, ou 64 kbps.

Na outra extremidade da chamada, um sinal analógico é recriado a partir das amostras quantizadas representando-as (e suavizando-as) com o tempo. Esse não será exatamente o mesmo que o sinal analógico original, embora tenhamos feito a amostragem na taxa Nyquist, pois as amostras foram quantizadas. Para reduzir o erro em virtude da quantização, os níveis desta são espaçados de forma desigual. Uma escala logarítmica é usada, gerando relativamente mais bits para menores amplitudes de sinal e relativamente menos bits para grandes amplitudes de sinal. Desse modo, o erro é proporcional à amplitude do sinal.

Duas versões de quantização são bastante utilizadas: **μ-law**, usada na América do Norte e no Japão, e **A-law**, usada na Europa e no restante do mundo. As duas versões são especificadas no padrão ITU G.711. Um modo equivalente de pensar nesse processo é imaginar que a faixa dinâmica do sinal (ou o rádio entre os maiores e menores valores possíveis) é compactada antes de ser quantizada (uniformemente), e então expandida quando o sinal analógico é recriado. Por esse motivo, ela é chamada **compactação/expansão**, ou **companding**. Também é possível compactar as amostras depois que elas são digitalizadas, de modo que exigem muito menos do que 64 kbps. Contudo, deixaremos esse assunto para quando explorarmos aplicações de áudio como VoIP.

Multiplexação por divisão de tempo

A TDM baseada na PCM é usada para transportar várias chamadas de voz por troncos, enviando uma amostra de cada chamada a cada 125 μs. Quando a transmissão digital começou a surgir como uma tecnologia viável, a ITU (também chamada CCITT) foi incapaz de chegar a um acordo sobre um padrão internacional para a PCM. Consequentemente, diversos esquemas incompatíveis agora estão em uso em diferentes países.

O método usado na América do Norte e no Japão é a portadora **T1**, representada na Figura 2.33. (Tecnicamente falando, o formato é chamado DS1 e a portadora é chamada T1, mas, seguindo a tradição da indústria, aqui não faremos essa distinção sutil.) A portadora T1 consiste em 24 canais de voz multiplexados. Por sua vez, cada um dos 24 canais consegue inserir 8 bits no fluxo de saída.

Um quadro consiste em $24 \cdot 8 = 192$ bits, mais um bit extra para fins de controle, produzindo 193 bits a cada 125 μs. Isso resulta em uma taxa de dados bruta de 1,544 Mbps, dos quais 8 kbps são para sinalização. O 193º bit é usado para sincronização e sinalização de quadros. Em uma variação, o 193º bit é usado por um grupo de 24 quadros, chamado **superquadro estendido**. Seis dos bits, na 4ª, 8ª, 12ª, 16ª, 20ª e 24ª posições, utilizam o padrão 001011... Normalmente, o receptor continua a conferir esse bit para

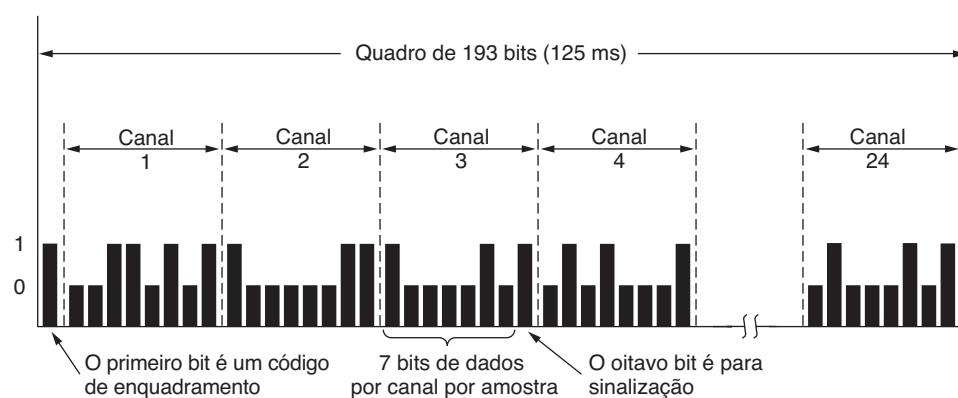



Figura 2.33 | A portadora T1 (1,544 Mbps).

garantir que não perdeu a sincronização. Seis outros bits são usados para enviar um código de verificação de erro, para ajudar o receptor a confirmar se está sincronizado. Se sair de sincronismo, o receptor poderá procurar por esse padrão e validar o código de verificação de erro para se ressincronizar. Os 12 bits restantes são usados para informação de controle, para operar e manter a rede, como o relatório de desempenho da extremidade remota.

O formato T1 tem diversas variações. As versões mais antigas enviavam informações de sinalização **na própria banda**, significando o uso do mesmo canal, usando alguns bits de dados. Esse projeto é uma forma de **sinalização associada ao canal**, pois cada canal tem seu próprio subcanal de sinalização privado. Em um arranjo de bits, o bit menos significativo de uma amostra de 8 em cada canal é usado a cada seis quadros. Este recebe o nome sugestivo de **bit de sinalização roubado**. A ideia é que alguns bits roubados não influenciarão nas chamadas de voz. Ninguém notará nenhuma diferença audível.

Contudo, para dados, a história é outra. Entregar os bits errados é, no mínimo, inútil. Se versões mais antigas da T1 forem usadas para transportar dados, somente 7 de 8 bits, ou 56 kbps, podem ser usados em cada um dos 24 canais. Em vez disso, versões mais novas da T1 oferecem canais limpos, em que todos os bits podem ser usados para enviar dados. Canais limpos são o que as empresas que alugam uma linha T1 desejam quando enviam dados pela rede telefônica no lugar das amostras de voz. A sinalização para quaisquer chamadas de voz é, então, tratada **fora da banda**, significando um canal próprio separado dos dados. Normalmente, a sinalização é feita com **sinalização de canal comum**, no qual existe um canal de sinalização compartilhado. Um dos 24 canais pode ser usado para essa finalidade.

Fora da América do Norte e do Japão, a portadora **E1** de 2,048 Mbps é usada no lugar da T1. Essa portadora tem 32 amostras de dados de 8 bits compactadas no quadro básico de 125 μs. Trinta dos canais são usados para informações e até dois são usados para sinalização. Cada grupo de quatro quadros oferece 64 bits de sinalização, metade deles usada para sinalização (associada ao canal ou ao canal comum) e metade usada para sincronização de quadros ou reservada para cada país usar como preferir.

A multiplexação por divisão de tempo permite que várias portadoras T1 sejam multiplexadas em portadoras de ordem mais alta. A Figura 2.34 mostra como isso pode ser feito. À esquerda, vemos quatro canais T1 sendo multiplexados em um canal T2. A multiplexação em T2 e acima é feita bit a bit, e não byte a byte, com os 24 canais de voz que compõem um quadro T1. Quatro fluxos T1 a uma velocidade de 1,544 Mbps deveriam gerar 6,176 Mbps, mas T2 na verdade tem 6,312 Mbps. Os bits extras são usados para enquadramento e recuperação, no caso de a portadora apresentar alguma falha. T1 e T3 são extensamente utilizados pelos clientes, enquanto T2 e T4 são usados apenas dentro do sistema de telefonia propriamente dito e, portanto, não são bem conhecidos.

No nível seguinte, sete fluxos T2 são combinados bit a bit para formar um fluxo T3. Depois, seis fluxos T3 são unidos para formar um fluxo T4. Em cada etapa, um pequeno volume de overhead é adicionado para fins de enquadramento e recuperação, no caso de a sincronização entre transmissor e receptor ser perdida.

Da mesma forma que existe pouco consenso quanto à portadora básica entre os Estados Unidos e o restante do mundo, há igualmente pouco consenso sobre como ela será multiplexada em portadoras de largura de banda mais alta. O esquema americano de avançar por 4, 7 e 6 não foi adotado por mais ninguém; assim, o padrão ITU requer multiplexação de quatro fluxos em um fluxo a cada nível. Além disso, o enquadramento e a recuperação de dados são diferentes entre os padrões dos Estados Unidos e da ITU. A hierarquia ITU para 32, 128, 512, 2.048 e 8.192 canais funciona em velocidades de 2,048, 8,848, 34,304, 139,264 e 565,148 Mbps.

SONET/SDH

Nos primórdios da fibra óptica, cada companhia telefônica tinha seu próprio sistema óptico TDM patenteado. Depois que a AT&T foi desmembrada, em 1984, as companhias telefônicas locais tiveram de se conectar a diversas concessionárias de comunicações de longa distância, todas com diferentes sistemas ópticos TDM, o que tornou óbvia a necessidade de padronização. Em 1985, a Bellcore, a unidade de pesquisa do RBOC, começou a trabalhar em um

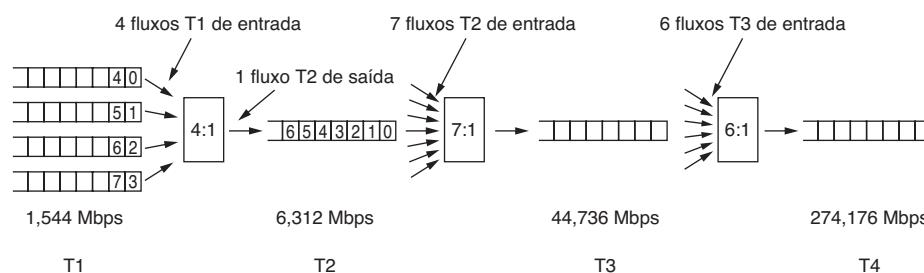



Figura 2.34 | Multiplexação de fluxos T1 em portadoras de velocidade mais alta.

padrão, denominado **rede óptica síncrona**, ou **SONET** (**Synchronous Optical NETwork**).

Mais tarde, a ITU também começou a participar desse trabalho, o que resultou em um padrão SONET e em um conjunto de recomendações paralelas da ITU (G.707, G.708 e G.709) em 1989. As recomendações da ITU são chamadas **hierarquia digital síncrona**, ou **SDH** (**Synchronous Digital Hierarchy**), mas diferem da SONET apenas em pequenos detalhes. Praticamente todo o tráfego telefônico de longa distância nos Estados Unidos e grande parte dele em outros lugares atualmente utiliza troncos que executam a SONET na camada física. Para obter informações adicionais sobre a SONET, consulte Bellamy (2000), Goralski (2002) e Shepard (2001).

O projeto SONET tem quatro objetivos principais. Acima de tudo, a SONET tinha de tornar possível a rede interligada para diferentes concessionárias. A concretização desse objetivo exigia a definição de um padrão de sinalização comum, relacionado a comprimento de onda, sincronização, estrutura de enquadramento e outras questões.

Em segundo lugar, foram necessários alguns meios para unificar os sistemas digitais de Estados Unidos, Europa e Japão, todos baseados em canais PCM de 64 kbps, mas todos combinados de formas diferentes (e incompatíveis).

Em terceiro lugar, a SONET teve de proporcionar um modo de multiplexar vários canais digitais. No momento em que a SONET surgiu, a portadora digital de velocidade mais alta usada em todo o território dos Estados Unidos era a T3, a 44,736 Mbps. A T4 já havia sido definida, mas não era muito usada, e nada que ultrapassasse a velocidade da T4 havia sido definido. Parte da missão da SONET era dar continuidade à hierarquia até gigabits/s e proporcionar velocidades ainda maiores. Também era necessária uma forma padrão de multiplexar canais mais lentos em um canal SONET.

Em quarto lugar, a SONET tinha de oferecer recursos de operação, administração e manutenção (OAM) necessários para gerenciar a rede. Os sistemas anteriores não faziam isso muito bem.

Uma decisão inicial foi tornar a SONET um sistema TDM tradicional, com toda a largura de banda da fibra dedicada a um único canal contendo slots de tempo para os diversos subcanais. Portanto, a SONET é um sistema síncrono. Cada transmissor e receptor é ligado a um clock comum. O relógio mestre, que controla o sistema, tem uma precisão de aproximadamente uma parte em 10^9. Os bits em uma linha SONET são transmitidos a intervalos extremamente precisos, controlados pelo clock mestre.

O quadro básico da SONET é um bloco de 810 bytes, transmitido a cada 125 μs. Tendo em vista que a SONET é síncrona, os quadros são emitidos independentemente de haver ou não dados úteis a enviar. A taxa de 8.000 quadros/s corresponde exatamente à taxa de amostragem dos canais PCM utilizados em todos os sistemas de telefonia digital.

Os quadros de 810 bytes da SONET são mais bem descritos como um retângulo de bytes, com 90 colunas de largura por 9 linhas de altura. Desse modo, $8 \times 810 = 6.480$ bits são transmitidos 8 mil vezes por segundo, o que resulta em uma taxa de dados bruta de 51,84 Mbps. Esse é o canal básico da SONET, chamado **STS-1** (**Synchronous Transport Signal-1**). Todos os troncos SONET são múltiplos do STS-1.

As três primeiras colunas de cada quadro são reservadas para as informações de gerenciamento do sistema, conforme ilustra a Figura 2.35. As três primeiras linhas contêm o overhead de seção; as seis linhas seguintes contêm o overhead de linha. O overhead de seção é gerado e verificado no início e no fim de cada seção, enquanto o overhead de linha é gerado e verificado no início e no fim de cada linha.

Um transmissor SONET transmite quadros de 810 bytes em sequência, sem intervalos entre eles, mesmo quando

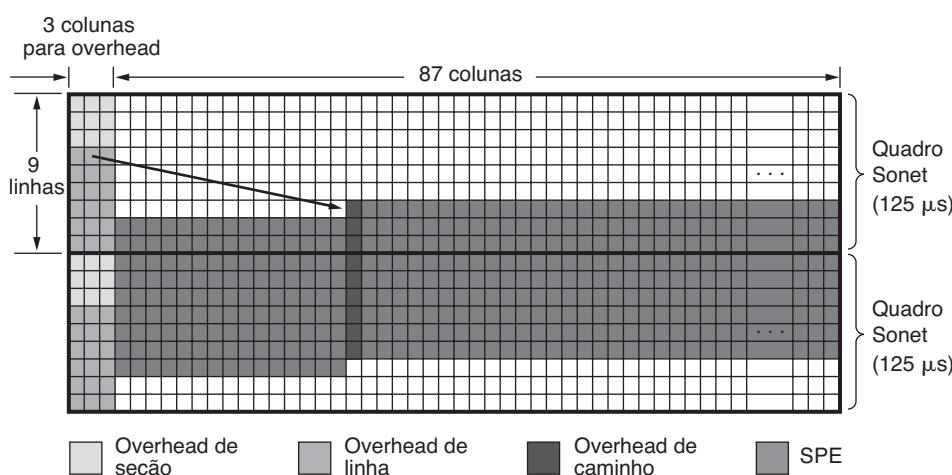



Figura 2.35 | Dois quadros duplos na rede SONET.

não existem dados (e, nesse caso, ele transmite dados fictícios). Do ponto de vista do receptor, tudo o que ele vê é um fluxo de bits contínuo; assim, como saber onde começa cada quadro? A resposta é que os dois primeiros bytes de cada quadro contêm um padrão fixo que o receptor procura. Se encontra esse padrão no mesmo lugar em um número grande de quadros consecutivos, o receptor pressupõe que está sincronizado com o transmissor. Em teoria, um usuário poderia inserir esse padrão como carga útil de uma maneira simples; porém, na prática, isso não pode ser feito em virtude da multiplexação usada por diversos usuários no mesmo quadro, além de outras razões.

As 87 colunas restantes de cada quadro contêm 87 × 9 × 8 × 8.000 = 50,112 Mbps de dados do usuário. A SONET é simplesmente um contêiner conveniente para transportar bits. O **envelope síncrono de carga útil**, ou **SPE** (**Synchronous Payload Envelope**), que transporta os dados do usuário, nem sempre começa na linha 1, coluna 4. O SPE pode começar em qualquer lugar do quadro. Um ponteiro para o primeiro byte está contido na primeira fileira do overhead de linha. A primeira coluna do SPE é o overhead de caminho (ou seja, o cabeçalho do protocolo da subcamada de caminho ponta a ponta).

A capacidade de permitir que o SPE comece em qualquer lugar dentro do quadro SONET e até mesmo se espalhe por dois quadros, como mostra a Figura 2.35, oferece flexibilidade adicional ao sistema. Por exemplo, se uma carga útil chega na origem enquanto um quadro SONET fictício está sendo construído, ele pode ser inserido no quadro atual em vez de ser mantido até o início do próximo.

A hierarquia de multiplexação da SONET/SDH é mostrada na Tabela 2.5. Foram definidas taxas de STS-1 a STS-768, variando de aproximadamente uma linha T3 até 40 Gbps. Até mesmo taxas maiores certamente serão definidas com o tempo, com OC-3072 a 160 Gbps sendo o próximo na fila, se e quando isso se tornar tecnologicamente viável. A portadora óptica que corresponde a STS-*n* é chamada OC-*n*, porém sua configuração bit a bit é a mesma, exceto por uma certa reordenação de bits, necessária para sincronização. Os nomes SDH são diferentes, começando em OC-3 porque os sistemas baseados na ITU não possuem uma taxa próxima a 51,84 Mbps. Mostramos as taxas comuns, que desenvolvem-se a partir de OC-3 em múltiplos de quatro. A taxa de dados bruta inclui todo o overhead. A taxa de dados SPE exclui o overhead de linha e o de seção. A taxa de dados do usuário exclui todo o overhead e só considera as 87 colunas de carga útil.

Por outro lado, quando uma portadora como a OC-3 não é multiplexada, mas transporta os dados de uma única origem, a letra *c* (significando concatenado) é acrescentada à designação; sendo assim, OC-3 indica uma portadora de 155,52 Mbps composta por três portadoras OC-1 distintas, mas OC-3c indica um fluxo de dados de uma única origem a uma velocidade de 155,52 Mbps. Os três fluxos OC-1 contidos em um fluxo OC-3c são entrelaçados por coluna: primeiro, a coluna 1 do fluxo 1, depois a coluna 1 do fluxo 2, a coluna 1 do fluxo 3, seguida pela coluna 2 do fluxo 1 e assim por diante, resultando em um quadro com 270 colunas de largura e 9 linhas de profundidade.

Multiplexação por divisão de comprimento de onda

Outra forma de multiplexação é usada tanto quanto a TDM para aproveitar a enorme largura de banda dos canais de fibra óptica. Trata-se da **multiplexação por divisão de comprimento de onda**, ou **WDM** (**Wavelength Division Multiplexing**). O princípio básico da WDM em fibras está representado na Figura 2.36. Aqui, quatro fibras chegam juntas a um colimador óptico, cada uma com sua energia presente em um comprimento de onda distinto. Os quatro feixes são combinados em uma única fibra compartilhada para transmissão a um destino remoto. Na extremidade remota, o feixe é dividido no mesmo número de fibras que havia no lado da entrada. Cada fibra de saída contém um núcleo curto, especialmente construído, que filtra todos os comprimentos de onda, com exceção de um. Os sinais resultantes podem ser roteados até seu destino ou recombinados de diferentes maneiras para transporte ou multiplexação adicional.

Realmente não há nada de novo aqui. Trata-se apenas da multiplexação por divisão da frequência em fre-

SONET		SDH	Taxa de dados (Mbps)		
Elétrico	Óptico	Óptico	Bruto	SPE	Usuário
STS-1	OC-1		51,84	50,112	49,536
STS-3	OC-3	STM-1	155,52	150,336	148,608
STS-12	OC-12	STM-4	622,08	601,344	594,432
STS-48	OC-48	STM-16	2.488,32	2.405,376	2.377,728
STS-192	OC-192	STM-64	9.953,28	9.621,504	9.510,912
STS-768	OC-768	STM-256	39.813,12	38.486,016	38.043,648

Tabela 2.5 | Taxas de multiplexação da SONET e da SDH.

quências muito altas, com o termo WDM devendo-se à descrição dos canais de fibra óptica em função de seu comprimento de onda característico em vez da frequência propriamente. Desde que cada canal tenha sua própria faixa de frequências (isto é, de comprimentos de onda) e todas as faixas sejam disjuntas, elas poderão ser multiplexadas na fibra de longa distância. A única diferença em relação à FDM elétrica é que um sistema óptico que utilize uma rede de difração será completamente passivo e, portanto, altamente confiável.

A razão para o WDM ser popular é o fato de a energia em uma única fibra normalmente ter apenas alguns gigahertz de largura porque, no momento, é impossível realizar a conversão entre sinais elétricos e ópticos com rapidez maior que essa. Utilizando-se muitos canais em paralelo com diferentes comprimentos de onda, a largura de banda agregada aumenta de forma linear com o número de canais. Como a largura de banda de uma única banda de fibra é de aproximadamente 25.000 GHz (veja a Figura 2.6), teoricamente existe espaço para 2.500 canais de 10 Gbps, mesmo a 1 bit/Hz (e também são possíveis taxas mais altas).

A tecnologia WDM tem progredido a uma taxa que faria a tecnologia computacional se envergonhar. O WDM foi criado por volta de 1990. Os primeiros sistemas comerciais tinham oito canais de 2,5 Gbps por canal. Em 1998, sistemas com 40 canais de 2,5 Gbps estavam no mercado. Em 2006, já eram utilizados produtos com 192 canais de 10 Gbps e 64 canais de 40 Gbps, capazes de mover até 2,56 Tbps. Essa largura de banda é suficiente para transmitir 80 filmes de DVD completos por segundo. Os canais também estão bastante compactados na fibra, com 200, 100 ou mesmo 50 GHz de separação. As demonstrações da tecnologia pelas empresas, após gabarem-se de seus feitos, têm mostrado dez vezes essa capacidade no laboratório, mas passar do laboratório para o campo normalmente leva pelo menos alguns anos. Quando o número de canais é muito grande e os comprimentos de onda são pouco espaçados, o sistema é chamado de **DWDM** (**Dense WDM**).

Um dos impulsionadores da tecnologia WDM é o desenvolvimento de amplificadores totalmente ópticos. Antes, a cada 100 km era necessário dividir todos os canais e converter cada um deles a um sinal elétrico para amplificação separada, antes de convertê-los novamente em sinais ópticos e combiná-los. Hoje, os amplificadores totalmente ópticos podem regenerar o sinal inteiro uma única vez a cada 1.000 km, sem a necessidade de várias conversões ópticas/elétricas.

No exemplo da Figura 2.36, temos um sistema de comprimento de onda fixo. Bits da fibra de entrada 1 vão para a fibra de saída 3, bits da fibra de entrada 2 vão para a fibra de saída 1 etc. Porém, também é possível criar sistemas WDM comutados no domínio óptico. Em dispositivos como esses, os filtros de saída são ajustáveis com o uso de interferômetros de Fabry-Perot ou de Mach-Zehnder. Esses dispositivos permitem que as frequências selecionadas sejam trocadas dinamicamente por um computador de controle. Essa capacidade oferece uma grande flexibilidade para a provisão de muitos caminhos de comprimento de onda diferentes através da rede telefônica a partir de um conjunto fixo de fibras. Para obter mais informações sobre redes ópticas e WDM, consulte Ramaswami et al. (2009).

2.6.5 Comutação

Do ponto de vista do engenheiro de telefonia, o sistema telefônico é dividido em duas partes principais: a planta externa (os circuitos terminais e troncos, pois eles estão localizados fisicamente fora das estações de comutação) e a planta interna (os switches, que estão dentro das estações

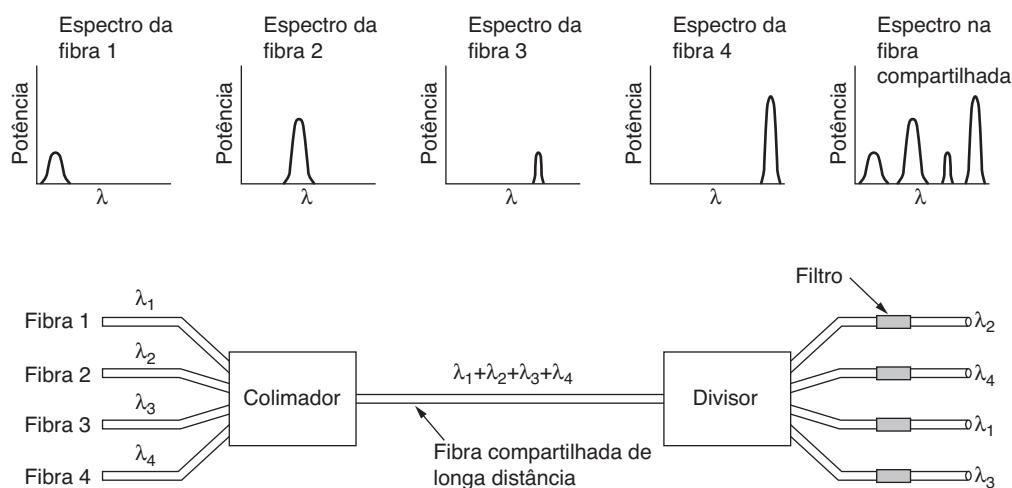



Figura 2.36 | Multiplexação por divisão de comprimento de onda.

de comutação). Acabamos de estudar a planta externa. Agora vamos examinar a planta interna.

Hoje em dia, duas técnicas de comutação diferentes são usadas pela rede: comutação de circuitos e comutação de pacotes. O sistema telefônico tradicional é baseado na comutação de circuitos, mas a comutação de pacotes está começando a ganhar terreno com o aumento da tecnologia VoIP. Veremos em detalhes a comutação de circuitos, comparando-a com a comutação de pacotes. Os dois tipos de comutação são tão importantes que voltaremos a eles quando entrarmos na camada de rede.

Comutação de circuitos

Conceitualmente, quando você ou seu computador efetuam uma chamada telefônica, o equipamento de comutação do sistema telefônico procura um caminho físico desde o seu telefone até o telefone do receptor. Essa técnica, chamada **comutação de circuitos**, é apresentada esquematicamente na Figura 2.37(a). Cada um dos seis retângulos representa uma estação de comutação da concessionária de comunicações (estação final, estação interurbana etc.). Nesse exemplo, cada estação tem três linhas de entrada e três linhas de saída. Quando uma chamada passa por uma estação de comutação, é (conceitualmente) estabelecida uma conexão física entre a linha que transportou a chamada e uma das linhas de saída, como mostram as linhas pontilhadas.

No início da telefonia, a conexão era feita pela telefonista que conectava um cabo de ligação em ponte (jumper) aos soquetes de entrada e saída. Na realidade, existe uma pequena história surpreendente associada à invenção do equipamento automático de comutação de circuitos. Esse dispositivo foi inventado por um agente funerário do século XIX, Almon B. Strowger. Logo depois que o telefone foi inventado, quando uma pessoa morria, alguém ligava para a telefonista da cidade e dizia: 'Por favor, ligue-me com um agente funerário'. Infelizmente para o sr. Strowger, havia dois agentes funerários em sua cidade, e a esposa do outro agente era a telefonista da cidade. Ele percebeu rapidamente que teria de inventar um equipamento automático de comutação telefônica ou seu negócio iria à falência. Ele escolheu a primeira opção. Por cerca de cem anos, o equipamento de comutação de circuitos usado em todo o mundo foi conhecido como **engrenagem de Strowger**. (A história não registra se a telefonista, desempregada, conseguiu emprego como operadora de informações, respondendo a perguntas como: 'Qual é o número do telefone do agente funerário?')

O modelo mostrado na Figura 2.37(a) é altamente simplificado, obviamente, porque partes do caminho físico entre os dois telefones podem de fato ser enlaces de micro-ondas ou de fibra, nos quais são multiplexadas milhares de chamadas. Entretanto, a ideia básica é válida: uma vez estabelecida uma chamada, haverá um caminho dedicado entre ambas as extremidades, e ele continuará a existir até que a chamada seja finalizada.

Uma propriedade importante da comutação de circuitos é a necessidade de se estabelecer um caminho ponta a

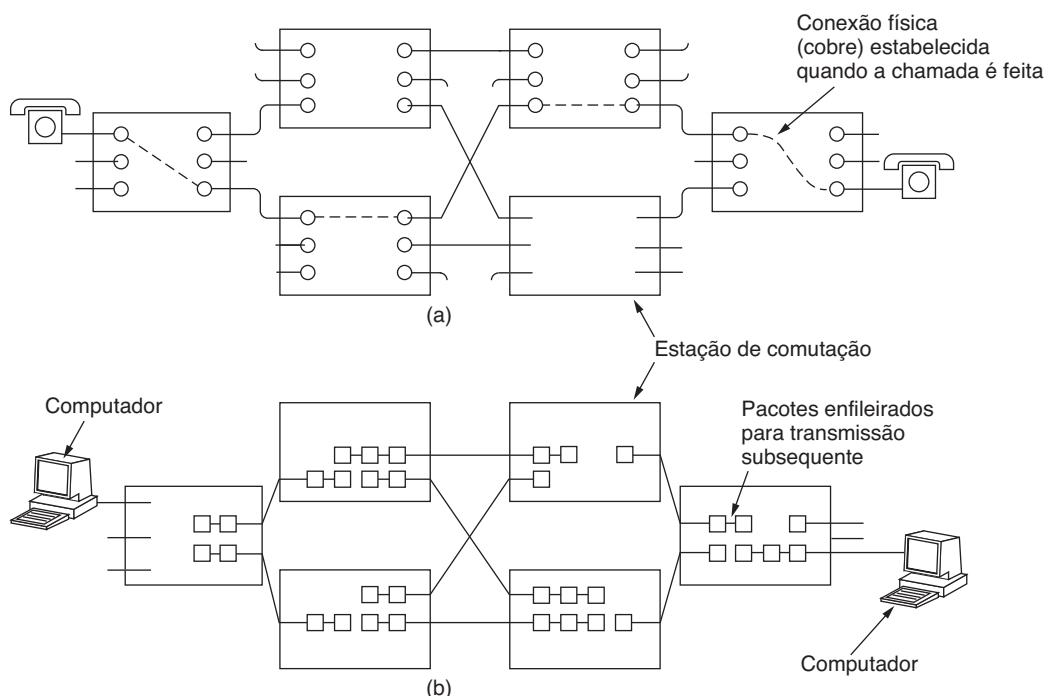



Figura 2.37 (a) Comutação de circuitos. (b) Comutação de pacotes.

ponta, *antes* que qualquer dado possa ser enviado. O tempo decorrido entre o fim da discagem e o momento em que o telefone começa a tocar pode chegar a 10 segundos ou mais em chamadas interurbanas ou internacionais. Durante esse intervalo, o sistema telefônico procura uma conexão física, como mostra a Figura 2.38(a). Observe que, antes mesmo de se iniciar a transmissão de dados, o sinal de solicitação de chamada deve se propagar em todo o trajeto até o destino e lá ser reconhecido. Para muitas aplicações de informática (por exemplo, a verificação de crédito em um ponto de venda), tempos de configuração longos são indesejáveis.

Como consequência do caminho reservado entre o transmissor e o receptor da chamada, uma vez estabelecida a configuração, o único atraso para a entrega dos dados é o tempo de propagação do sinal eletromagnético, cerca de 5 ms por 1.000 km. Outra consequência do caminho estabelecido é que não há perigo de congestionamento — ou seja, quando a chamada é feita, você nunca obtém sinal de ocupado. É claro que é possível receber um sinal de ocupado antes do estabelecimento da conexão, em decorrência da falta de capacidade de comutação ou de troncos.

Comutação de pacotes

A alternativa à comutação de circuitos é a **comutação de pacotes**, mostrada na Figura 2.37(b) e descrita no Capítulo 1. Com essa tecnologia, os pacotes são enviados assim que estão disponíveis. Não é preciso estabelecer um caminho dedicado com antecedência, diferente da comutação de circuitos. Fica a critério dos roteadores usar a transmissão store-and-forward para enviar cada pacote em seu caminho ao destino por conta própria. Esse procedimento é diferente da comutação de circuitos, em que o resultado do estabelecimento da conexão é a reserva de largura de banda desde o transmissor até o receptor. Todos os dados no circuito seguem esse caminho. Entre outras propriedades, fazer todos os pacotes seguirem o mesmo caminho significa que eles não poderão chegar fora de ordem. Com a comutação de pacotes, não há nenhum caminho fixo e, assim, diferentes pacotes podem seguir caminhos distintos, dependendo das condições da rede no momento em que eles são enviados. Portanto, eles podem chegar fora de ordem.

As redes de comutação de pacotes impõem um limite superior apertado sobre o tamanho dos pacotes. Isso garante que nenhum usuário poderá monopolizar qualquer linha de transmissão por muito tempo (por exemplo, muitos milissegundos), de modo que as redes de comutação de pacotes podem lidar com o tráfego interativo. Isso também reduz o atraso, pois o primeiro pacote de uma mensagem longa pode ser encaminhado antes que o segundo tenha chegado por completo. Contudo, o atraso

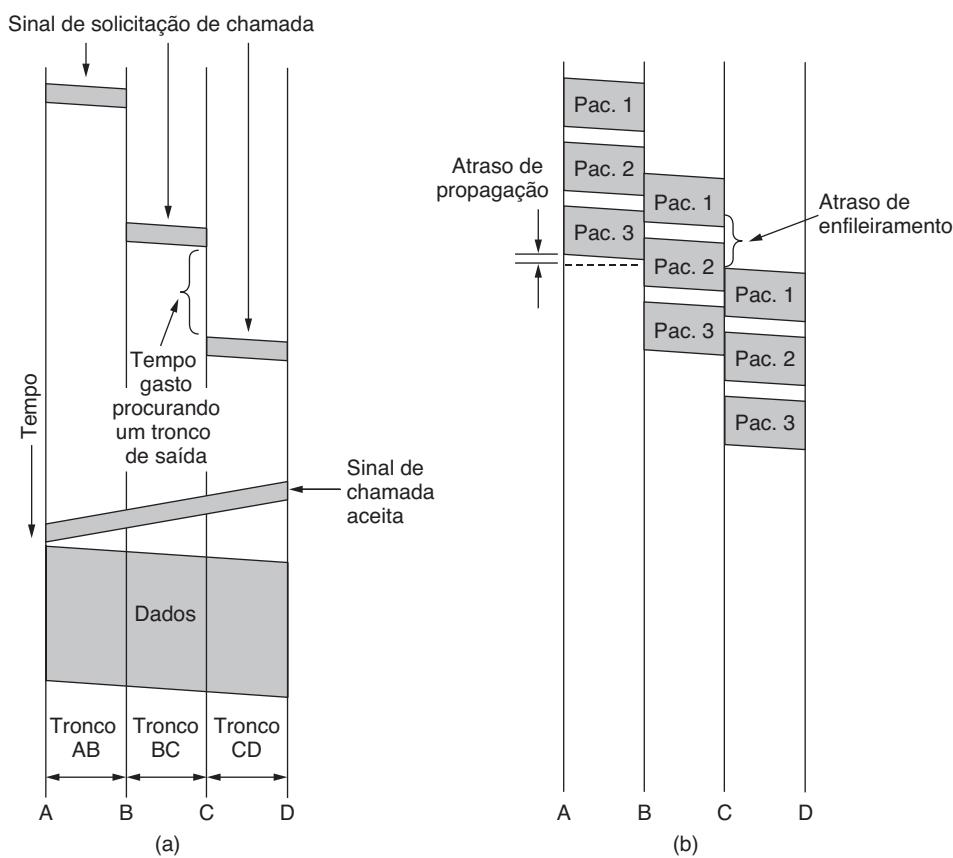



Figura 2.38 | Sincronização de eventos em (a) comutação de circuitos, (b) comutação de pacotes.

store-and-forward de acumular um pacote na memória do roteador antes que ele seja enviado para o próximo roteador excede o da comutação de circuitos. Com a comutação de circuitos, os bits simplesmente fluem pelo fio de modo contínuo.

A comutação de pacotes e de circuitos também difere de outras maneiras. Como nenhuma largura de banda é reservada com a comutação de pacotes, estes podem ter de esperar para serem encaminhados. Se muitos pacotes forem enviados ao mesmo tempo, isso introduz o **atraso de enfileiramento** e o congestionamento. Por outro lado, não existe perigo de obter um sinal de ocupado e não conseguir usar a rede. Assim, o congestionamento ocorre em diferentes momentos com a comutação de circuitos (no momento da configuração) e a comutação de pacotes (quando os pacotes são enviados).

Se um circuito tiver sido reservado para determinado usuário e não houver tráfego, sua largura de banda é desperdiçada. Ela não pode ser usada para outro tráfego. A comutação de pacotes não desperdiça largura de banda e, portanto, é mais eficiente do ponto de vista do sistema. Entender essa escolha é decisivo para compreender a diferença entre comutação de circuitos e comutação de pacotes. O dilema está entre garantir serviço e desperdiçar recursos *versus* não garantir serviço e não desperdiçar recursos.

A comutação de pacotes é mais tolerante a falhas que a comutação de circuitos. De fato, é por isso que ela foi criada. Se um switch ficar inativo, todos os circuitos que o utilizam serão encerrados, e nenhum tráfego poderá mais ser transmitido em qualquer um deles. Com a comutação de pacotes, os pacotes poderão ser roteados de modo a contornar switches inativos.

Uma última diferença entre a comutação de circuitos e a de pacotes é o algoritmo de tarifação. Com a comutação de circuitos, a tarifação se baseava historicamente na distância e no tempo. No caso dos telefones móveis, em geral a distância não é importante, exceto para chamadas internacionais, e o tempo desempenha apenas um papel secundário (por exemplo, um plano de chamadas com 2 mil minutos gratuitos custa mais que um plano com mil minutos gratuitos e, algumas vezes, chamadas noturnas ou nos finais de semana são mais econômicas que o normal). Com a comutação de pacotes, o tempo de conexão não é um problema, mas o volume de tráfego sim. Para usuários domésticos, os ISPs normalmente cobram uma tarifa fixa mensal, porque tal modelo é menos trabalhoso para eles e mais fácil de entender para os clientes, mas as concessionárias de backbone cobram das redes regionais com base no volume de seu tráfego.

As diferenças estão resumidas na Tabela 2.6. Tradicionalmente, as redes telefônicas têm usado a comutação de circuitos para oferecer chamadas telefônicas de alta qualidade, e as redes de computadores têm usado a comutação de pacotes por suas simplicidade e eficiência. Contudo, existem exceções dignas de nota. Algumas redes de computadores mais antigas têm sido comutadas por circuitos 'por debaixo dos panos' (por exemplo, X.25), e algumas redes telefônicas mais novas usam a comutação de pacotes com a tecnologia VoIP. Para os usuários, isso se parece externamente com uma chamada telefônica padrão, mas, internamente, os pacotes na rede com dados de voz são comutados. Com essa técnica, as chamadas internacionais com cartões de chamada podem se tornar mais baratas, embora talvez com uma qualidade de chamada inferior à do serviço tradicional.

Item	Comutação de circuitos	Comutação de pacotes
Configuração de chamadas	Obrigatória	Não necessária
Caminho físico dedicado	Sim	Não
Cada pacote segue a mesma rota	Sim	Não
Os pacotes chegam em ordem	Sim	Não
A falha de um switch é fatal	Sim	Não
Largura de banda disponível	Fixa	Dinâmica
Momento de possível congestionamento	Durante a configuração	Em todos os pacotes
Largura de banda potencialmente desperdiçada	Sim	Não
Transmissão store-and-forward	Não	Sim
Tarifação	Por minuto	Por pacote

Tabela 2.6 | Uma comparação entre redes de comutação de circuitos e redes de comutação de pacotes.

2.7 O sistema de telefonia móvel

O sistema telefônico tradicional (ainda que algum dia ele chegue a vários gigabits entre uma extremidade e outra da fibra) não será capaz de satisfazer a um grupo crescente de usuários: as pessoas em trânsito. Agora, elas esperam efetuar chamadas telefônicas e usar seus telefones para verificar e-mail e navegar pela Web em aviões, carros, piscinas e enquanto caminham no parque. Consequentemente, há um enorme interesse na telefonia sem fios. Nas próximas seções, estudaremos esse tópico em detalhes.

O sistema de telefonia móvel é usado para comunicação remota de voz e dados. Os **telefones móveis** (às vezes chamados **telefones celulares**) passaram por três gerações distintas, normalmente chamadas **1G**, **2G** e **3G**. As gerações são:

1. Voz analógica.
2. Voz digital.
3. Voz digital e dados (Internet, correio eletrônico etc.).

(Os telefones móveis não devem ser confundidos com **telefones sem fio**, que consistem em uma estação-base e um aparelho móvel, vendidos como um conjunto para uso dentro da casa. Tais telefones nunca foram usados para redes, de modo que não os veremos mais a fundo.)

Embora a maior parte de nossa discussão seja sobre a tecnologia desses sistemas, é interessante observar como pequenas decisões políticas e de marketing podem ter um enorme impacto. O primeiro sistema móvel foi criado nos Estados Unidos pela AT&T e regulamentado para todo o país pela FCC. Como resultado, todo o território dos Estados Unidos tinha um único sistema (analógico), e um telefone móvel adquirido na Califórnia também funcionava em Nova York. Ao contrário, quando a tecnologia chegou à Europa, cada país criou seu próprio sistema, o que resultou em um fiasco.

A Europa aprendeu com seus erros e, ao surgir a tecnologia digital, as PTTs estatais se juntaram e padronizaram um único sistema (GSM); portanto, qualquer telefone móvel europeu funcionará em qualquer lugar da Europa. Na época, os Estados Unidos haviam decidido que o governo não deveria participar do esforço de padronização, e assim a padronização da tecnologia digital ficou a cargo do mercado. Essa decisão resultou em diferentes fabricantes de equipamentos produzindo tipos distintos de telefones móveis. Em consequência disso, os Estados Unidos agora têm dois importantes sistemas de telefonia móvel digital totalmente incompatíveis em operação (além de outros sistemas secundários).

Apesar da liderança inicial dos Estados Unidos, a propriedade e a utilização da telefonia móvel na Europa é agora muito maior que naquele país. O fato de haver um único sistema para toda a Europa explica em parte esse fato, mas há outras razões. Um segundo ponto em que os Estados Unidos e a Europa divergiram foi a questão dos números de telefone. Nos Estados Unidos, os telefones móveis têm números misturados com os telefones comuns (fixos). Desse modo, quem faz a ligação não tem como saber se, digamos, (212) 234-5678 é um telefone fixo (com uma ligação de baixo custo ou gratuita) ou um telefone móvel (com uma tarifa cara). Para impedir que as pessoas ficassem receosas de usar o telefone, as empresas de telefonia decidiram fazer o proprietário do telefone móvel pagar pelas chamadas recebidas. Em consequência disso, muitas pessoas hesitaram em comprar um telefone móvel por medo de terem de pagar uma conta enorme apenas por receberem ligações. Na Europa, os telefones móveis têm um código de área especial (semelhante aos números 800 e 900) e, assim, podem ser reconhecidos instantaneamente. Como resultado, a regra habitual de 'fazer o chamador pagar' também se aplica aos telefones móveis da Europa (com exceção das ligações internacionais, cujos custos são divididos).

Uma terceira questão que teve grande impacto na adoção da telefonia móvel foi o uso difundido de telefones pré-pagos na Europa (até 75 por cento em algumas regiões). Esses telefones podem ser adquiridos em muitas lojas sem mais formalidades que a compra de uma câmera fotográfica digital: você paga e leva. Eles são pré-carregados com, por exemplo, 20 ou 50 euros em ligações e podem ser recarregados (com a utilização de um código PIN secreto) quando o saldo cai a zero. Por essa razão, praticamente todos os adolescentes e muitas crianças pequenas na Europa têm telefones móveis (em geral, pré-pagos) para que seus pais possam localizá-los, sem o perigo de terem de pagar uma conta enorme. Se o telefone móvel for usado apenas ocasionalmente, seu uso será quase gratuito, pois não haverá tarifa mensal nem por chamadas recebidas.

2.7.1 Telefones móveis de primeira geração (1G): voz analógica

Já estudamos os aspectos políticos e de marketing dos telefones móveis. Agora, vamos examinar a tecnologia, começando pelo sistema mais antigo. Os radiotelefones móveis eram usados esporadicamente na comunicação militar e marítima, durante as primeiras décadas do século XX. Em 1946, foi criado em St. Louis, nos Estados Unidos, o primeiro sistema para telefones baseados em automóveis. O sistema utilizava um único transmissor grande no topo de um alto edifício e tinha um único canal, usado para transmissões e recepções. Para conversar, o usuário tinha de apertar um botão que ativava o transmissor e desativava o receptor. Tais sistemas, conhecidos como **sistemas 'apertar para falar'** (**push-to-talk systems**), foram instalados em diversas cidades a partir da década de 1950. Sistemas de radioamador, táxis e carros de polícia utilizam essa tecnologia com frequência.

Na década de 1960, o **sistema de telefonia móvel aperfeiçoado**, ou **IMTS** (**Improved Mobile Telephone System**) foi instalado. Ele também utilizava um transmis-

sor de alta potência (200 watts) no topo de uma montanha, mas agora tinha duas frequências: uma para transmissão e outra para recepção. Por isso, o botão 'apertar para falar' não era mais necessário. Como toda a comunicação dos telefones móveis utilizava um canal para transmissão e outro para recepção dos sinais, os usuários móveis não podiam ouvir uns aos outros (ao contrário do que acontecia com o sistema utilizado em táxis).

O IMTS admitia 23 canais espalhados pelas frequências de 150 a 450 MHz. Em virtude do pequeno número de canais, muitas vezes os usuários tinham de esperar muito tempo antes de obter um tom de discagem. Além disso, pela alta potência do transmissor, os sistemas adjacentes tinham de estar a vários quilômetros de distância uns dos outros para evitar interferência. Em suma, sua capacidade limitada tornou o sistema impraticável.

AMPS (Advanced Mobile Phone System)

Tudo isso mudou com o **sistema avançado de telefonia móvel**, ou **AMPS** (**Advanced Mobile Phone System**), inventado pelo Bell Labs e instalado inicialmente nos Estados Unidos em 1982. Ele também foi usado na Inglaterra, onde recebeu o nome TACS, e no Japão, onde foi chamado MCS-L1. O uso do AMPS foi encerrado formalmente em 2008, mas vamos examiná-lo para entender o contexto para os sistemas 2G e 3G, que o aperfeiçoaram.

Em todos os sistemas de telefonia móvel, uma região geográfica é dividida em **células**, e é esse o motivo pelo qual esses dispositivos são chamados telefones celulares. No AMPS, as células em geral têm de 10 a 20 km; nos sistemas digitais, as células são menores. Cada célula utiliza algum conjunto de frequências não utilizado por qualquer uma das células vizinhas. A ideia fundamental que dá aos sistemas celulares uma capacidade muito maior que a dos sistemas anteriores é o uso de células relativamente pequenas e a reutilização de frequências de transmissão em células próximas (mas não adjacentes). Enquanto um sistema IMTS com um alcance de 100 km pode ter uma chamada em cada frequência, um sistema AMPS pode ter cem células de 10 km na mesma região e é capaz de estabelecer de 5 a 10 chamadas em cada frequência, em células amplamente afastadas. Portanto, a estrutura celular aumenta a capacidade do sistema em pelo menos uma ordem de grandeza ou mais, se as células forem menores. Além disso, células menores significam menor necessidade de energia, o que possibilita a existência de dispositivos transmissores e receptores menores e mais econômicos.

A ideia de reutilização de frequências é ilustrada na Figura 2.39(a). Em geral, as células são razoavelmente circulares; porém, é mais simples representá-las como hexágonos. Na Figura 2.39(a), todas as células têm o mesmo tamanho. Elas são agrupadas em unidades de sete células. Cada letra indica um grupo de frequências. Observe que, para cada conjunto de frequências, existe um afastamento de aproximadamente duas células de extensão, no qual essa frequência não é reutilizada, o que proporciona boa separação e pouca interferência.

Encontrar locais altos para colocar antenas de estação-base é uma questão fundamental. Esse problema levou algumas concessionarias de telecomunicações a fazer alianças com a Igreja Católica Romana, que possui um número significativo de locais apropriados para a instalação de antenas em todo o mundo, todos convenientemente controlados por uma única administração.

Em uma área em que o número de usuários cresce a ponto de o sistema ficar sobrecarregado, a potência pode ser reduzida, e as células sobrecarregadas são divididas em células menores, chamadas **microcélulas**, para permitir maior reutilização de frequências, como mostra a Figura 2.39(b). Algumas vezes, as empresas de telefonia criam microcélulas temporárias, utilizando torres portáteis com enlaces de satélite para atender à demanda de eventos

(a)

(b)

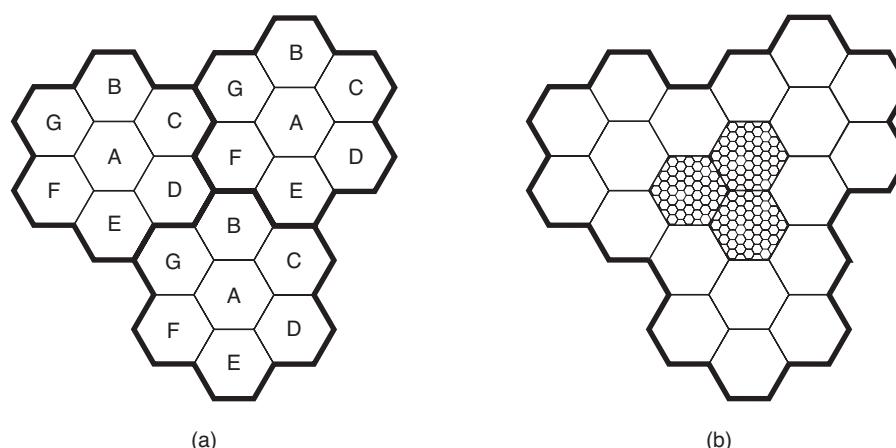



Figura 2.39 | (a) As frequências não são reutilizadas nas células adjacentes. (b) Para aumentar o número de usuários, podem ser utilizadas células menores.

esportivos, shows de rock e outros eventos nos quais um grande número de usuários de telefones celulares se reúne por algumas horas.

No centro de cada célula há uma estação-base que recebe as transmissões de todos os telefones presentes na célula. Essa estação consiste em um computador e um transmissor/receptor, conectados a uma antena. Em um sistema de pequeno porte, todas as estações-base estão conectadas a um único dispositivo, chamado **centro de comutação móvel**, ou **MSC** (**Mobile Switching Center**). Ele também é conhecido como **estação de comutação de telefonia móvel**, ou **MTSO** (**Mobile Telephone Switching Office**). Em um sistema maior, podem ser necessários vários MSCs, todos conectados a um MSC de segundo nível, e assim por diante. Basicamente, os MSCs são estações finais, como no sistema telefônico, e na verdade estão conectadas a pelo menos uma estação final de um sistema telefônico. Os MSCs se comunicam com as estações-base, entre si e com a PSTN usando uma rede de comutação de pacotes.

Em qualquer instante, cada telefone móvel logicamente ocupa uma célula específica e está sob o controle da estação-base dessa célula. Quando um telefone móvel deixa fisicamente uma célula, sua estação-base detecta que o sinal do telefone está enfraquecendo e questiona todas as estações-base vizinhas quanto à quantidade de energia que elas estão recebendo dele. Quando as respostas retornam, a estação-base faz a transferência para a célula que está obtendo o sinal mais forte; quase sempre, essa é a célula em que o telefone está localizado no momento. O telefone é, então, informado de quem é o seu novo chefe e, se houver uma chamada em andamento, ele será solicitado a passar para um novo canal (porque o antigo não é reutilizado em nenhuma das células adjacentes). Esse processo é chamado **handoff**, e leva cerca de 300 ms. A atribuição de canais é feita pelo MSC, o centro nervoso do sistema. Na verdade, as estações-base são apenas simples retransmissoras de rádio.

Canais

O AMPS utiliza FDM para separar os canais. O sistema utiliza 832 canais full-duplex, cada um consistindo em um par de canais simplex. Esse arranjo é conhecido como **duplex por divisão de frequência**, ou **FDD** (**Frequency Division Duplex**). Os 832 canais simplex de 824 a 849 MHz são usados para transmissão do aparelho móvel à estação-base, e 832 canais simplex de 869 a 894 MHz são usados para transmissão da estação-base ao aparelho móvel. Cada um desses canais simplex tem 30 kHz de largura de banda.

Os 832 canais estão divididos em quatro categorias. Canais de controle (da base para a unidade móvel) são usados para gerenciar o sistema. Canais de localização (da base para a unidade móvel) alertam os usuários móveis a respeito de chamadas destinadas a eles. Canais de acesso (bidirecionais) são usados para estabelecimento de chamadas e atribuição de canais. Por fim, os canais de dados (bidirecionais) transportam voz, fax ou dados. Como as mesmas frequências não podem ser reutilizadas em células vizinhas e 21 desses canais são reservados em cada célula para controle, o número real de canais de voz disponíveis por célula é bem menor que 832; normalmente está em torno de 45.

Gerenciamento de chamadas

Cada telefone móvel tem um número de série de 32 bits e um número de telefone de dez dígitos em sua PROM (memória programável somente de leitura). O número de telefone é representado como um código de área de 3 dígitos em 10 bits e um número de assinante de 7 dígitos em 24 bits. Quando um telefone é contactado, varre uma lista pré-programada de 21 canais de controle até encontrar o sinal mais forte. Em seguida, o telefone transmite seu número de série de 32 bits e o número de telefone de 34 bits. A exemplo de todas as outras informações de controle do AMPS, esse pacote é enviado várias vezes em formato digital e com um código de correção de erros, apesar de os próprios canais de voz serem analógicos.

Quando ouve a mensagem, a estação-base avisa ao MSC, que registra a existência de seu novo cliente e também informa a localização atual do cliente ao MSC local. Durante a operação normal, o telefone móvel repete o registro uma vez a cada 15 minutos, aproximadamente.

Para fazer uma chamada, o usuário móvel liga o telefone, digita no teclado o número a ser chamado e pressiona o botão SEND. Em seguida, o telefone transmite o número a ser chamado e sua própria identidade no canal de acesso. Se houver uma colisão, ele tenta novamente mais tarde. Ao receber a solicitação, a estação-base informa ao MSC. Se o chamador for um cliente da empresa do MSC (ou de uma de suas parceiras), o MSC procura um canal disponível para a chamada. Se encontrar algum, o número do canal será enviado de volta no canal de controle. Em seguida, o telefone móvel se conecta automaticamente ao canal de voz selecionado e aguarda até que a parte chamada atenda ao telefone.

As chamadas recebidas funcionam de forma diferente. Para começar, todos os telefones inativos ouvem continuamente o canal de localização para detectar as mensagens destinadas a eles. Quando é feita uma chamada para um telefone móvel (a partir de um telefone fixo ou de outro telefone móvel), um pacote é enviado ao MSC local do telefone chamado, para que ele seja localizado. Em seguida, é enviado um pacote à estação-base em sua célula atual, que, então, envia um pacote de difusão no canal de localização com o formato: 'Unidade 14, você esta aí?'. O telefone chamado responde 'Sim' no canal de acesso. Depois, a base transmite algo como: 'Unidade 14, chamada para você no canal 3'. Nesse momento, o telefone chamado se conecta ao canal 3 e começa a emitir sinais sonoros (ou a tocar alguma melodia que o proprietário do telefone ganhou como presente de aniversário).

2.7.2 Telefones móveis de segunda geração (2G): voz digital

A primeira geração de telefones celulares era analógica; a segunda geração é digital. A troca para digital tem diversas vantagens. Ela oferece ganhos de capacidade, permitindo que os sinais de voz sejam digitalizados e compactados. Ela melhora a segurança, permitindo que sinais de voz e de controle sejam criptografados. Isso, por sua vez, impede fraude e bisbilhotagem, seja por varredura intencional, seja por ecos de outras chamadas, em virtude da propagação de RF. Por fim, ela capacita novos serviços, como mensagens de texto.

Assim como não havia padronização internacional durante a primeira geração, também não havia padronização internacional durante a segunda. Vários sistemas diferentes foram desenvolvidos, e três foram largamente implementados. O **sistema avançado de telefonia móvel digital**, ou **D-AMPS** (**Digital Advanced Mobile Phone System**) é uma versão digital do AMPS que coexiste com AMPS e usa TDM para fazer várias chamadas no mesmo canal de frequência. Ele é descrito no padrão internacional IS-54 e seu sucessor, o IS-136. O **sistema global para comunicações móveis**, ou **GSM** (**Global System for Mobile communications**) apareceu como o sistema dominante, e, embora tenha demorado para ser aceito nos Estados Unidos, agora é usado praticamente em todo o mundo. Assim como o D-AMPS, o GSM é baseado em uma mistura de FDM e TDM. O **acesso múltiplo por divisão de código**, ou **CDMA** (**Code Division Multiple Access**), descrito no **padrão internacional IS-95**, é um tipo de sistema completamente diferente, que não é baseado nem em FDM nem em TDM. Embora o CDMA não tenha se tornado o sistema 2G dominante, sua tecnologia se tornou a base para os sistemas 3G.

Além disso, o nome **serviços de comunicações pessoais**, ou **PCS** (**Personal Communications Services**) às vezes é usado na literatura de marketing para indicar um sistema de segunda geração (ou seja, digital). Inicialmente, isso indicava um telefone móvel usando a banda de 1.900 MHz, mas essa distinção raramente é feita nos dias atuais.

Agora vamos descrever o GSM, pois é o sistema 2G dominante. Na próxima seção, quando descrevermos os sistemas 3G, falaremos mais sobre CDMA.

GSM (Global System for Mobile communications)

O GSM surgiu na década de 1980 como um esforço para produzir um único padrão 2G europeu. A tarefa foi atribuída a um grupo de telecomunicações chamado (em francês) Groupe Specialé Mobile. Os primeiros sistemas GSM foram implantados a partir de 1991 e experimentaram um sucesso repentino. Logo, ficou claro que o GSM seria mais do que um sucesso europeu, sendo absorvido até mesmo em países como a Austrália, de modo que o sistema foi renomeado para que tivesse um apelo mais global.

O GSM e outros sistemas de telefonia móvel que estudaremos retêm, dos sistemas 1G, um projeto baseado em células, reutilização de frequência pelas células e mobilidade com handoffs à medida que os assinantes se movem. São os detalhes que diferem. A seguir, descreveremos algumas das principais propriedades do GSM. Entretanto, o padrão GSM impresso tem mais de 5.000 [sic] páginas. Uma grande fração desse material se relaciona aos aspectos de engenharia do sistema, em especial ao projeto dos receptores para tratar da propagação de sinais por vários caminhos, e a sincronização de transmissores e receptores. Nada disso será sequer mencionado aqui.

A Figura 2.40 mostra que a arquitetura do GSM é semelhante à arquitetura do AMPS, embora os componentes tenham nomes diferentes. O próprio aparelho móvel agora é dividido em um aparelho e em um chip removível, com informações do assinante e da conta, chamado **cartão SIM**, uma abreviação de **Subscriber Identity Module** (**módulo de identidade do assinante**). É o cartão SIM que ativa o aparelho e contém segredos que permitem que o aparelho e a rede se identifiquem e codifiquem as conversas. Um cartão SIM pode ser removido e colocado em um aparelho diferente, para que este se torne seu aparelho móvel em relação à rede.

O telefone móvel fala com as estações-base da célula por uma **interface com o ar**, que descreveremos em breve. As estações-base da célula estão conectadas a um **controlador de estação-base**, ou **BSC** (**Base Station Controller**), que controla os recursos de rádio das células e cuida do handoff. O BSC, por sua vez, está conectado a um MSC (como no AMPS), que direciona as chamadas e as conecta à **rede de telefonia pública comutada**, ou **PSTN** (**Public Switched Telephone Network**).

Para poder direcionar as chamadas, o MSC precisa saber onde os aparelhos podem ser encontrados atualmente. Ele mantém um banco de dados dos aparelhos nas vizinhanças, que estão associados às células que ele controla. Esse banco de dados é chamado **registrador de local do visitante**, ou **VLR** (**Visitor Location Register**). Também há um banco de dados na rede móvel que indica o último local conhecido de cada aparelho, chamado **registrador de local inicial**, ou **HLR** (**Home Location Register**). Esse banco de dados é usado para direcionar as chamadas que chegam para os locais corretos. Os dois bancos de dados devem ser mantidos atualizados enquanto os aparelhos passam de uma célula para outra.

Agora, vamos descrever a interface com o ar em alguns detalhes. O GSM trabalha em uma faixa de frequências internacional, incluindo 900, 1.800 e 1.900 MHz. Foi alocado um espectro maior que o AMPS, para dar suporte a um número muito maior de usuários. O GSM é um sistema celular duplex por divisão de frequência, como o AMPS. Ou seja, cada aparelho móvel transmite em uma frequência e recebe em outra mais alta (55 MHz mais alta para GSM,

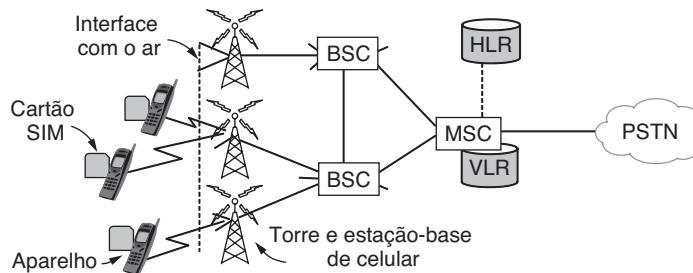



Figura 2.40 | Arquitetura móvel da rede GSM.

contra 80 MHz mais alta para AMPS). Contudo, diferente do AMPS, com o GSM, um único par de frequências é dividido pela multiplexação por divisão em slots de tempo. Desse modo, ele é compartilhado por vários aparelhos.

Para lidar com vários aparelhos, os canais GSM são muito mais largos que os canais AMPS (200 kHz contra 30 kHz). Um canal de 200 kHz é representado na Figura 2.41. Um sistema GSM operando na região de 900 MHz tem 124 pares de canais simplex. Cada canal simplex tem 200 kHz de largura e aceita oito conexões separadas nele, usando a multiplexação por divisão de tempo. Cada estação atualmente ativa recebe um slot de tempo em um par de canais. Teoricamente, 992 canais podem ser aceitos em cada célula, mas muitos deles não estão disponíveis, para evitar conflitos de frequência com células vizinhas. Na Figura 2.41, todos os oito slots de tempo sombreados pertencem à mesma conexão, quatro deles em cada direção. A transmissão e a recepção não acontecem no mesmo slot de tempo, pois os rádios GSM não podem transmitir e receber ao mesmo tempo, e leva algum tempo para passar de um para o outro. Se a unidade móvel atribuída à faixa de 890,4/935,4 MHz e ao slot de tempo 2 quisesse transmitir algo para a estação-base, ela usaria os quatro slots sombreados inferiores (e os slots depois deles no tempo), inserindo alguns dados em cada slot até que todos os dados fossem enviados.

Os slots TDM mostrados na Figura 2.41 fazem parte de uma complexa hierarquia de enquadramento. Cada slot TDM tem uma estrutura específica, e grupos de slots TDM formam multiquadros, também com uma estrutura específica. Uma versão simplificada dessa hierarquia é mostrada na Figura 2.42. Aqui, podemos ver que cada slot TDM consiste em um quadro de dados de 148 bits que ocupa o canal por 577 μs (incluindo um tempo de proteção de 30 μs depois de cada slot). Cada quadro de dados começa e termina com três bits 0, para fins de delineação de quadros. Ele também contém dois campos *Informação* de 57 bits, cada um com um bit de controle que indica se o campo *Informação* seguinte se refere a voz ou a dados. Entre os campos *Informação* há um campo (de treinamento) *Sync* de 26 bits, usado pelo receptor para realizar a sincronização até os limites de quadro do transmissor.

Um quadro de dados é transmitido em 547 μs, mas um transmissor só pode enviar um quadro de dados a cada 4,615 ms, pois ele está compartilhando o canal com sete outras estações. A taxa bruta de cada canal é de 270.833 bps, dividida entre oito usuários. Porém, como ocorre com o AMPS, o overhead consome uma grande fração da largura de banda, deixando em última análise 24,7 kbps de carga útil por usuário antes da correção de erros. Após essa correção, restam 13 kbps para voz. Embora isso seja substancialmente menor que os

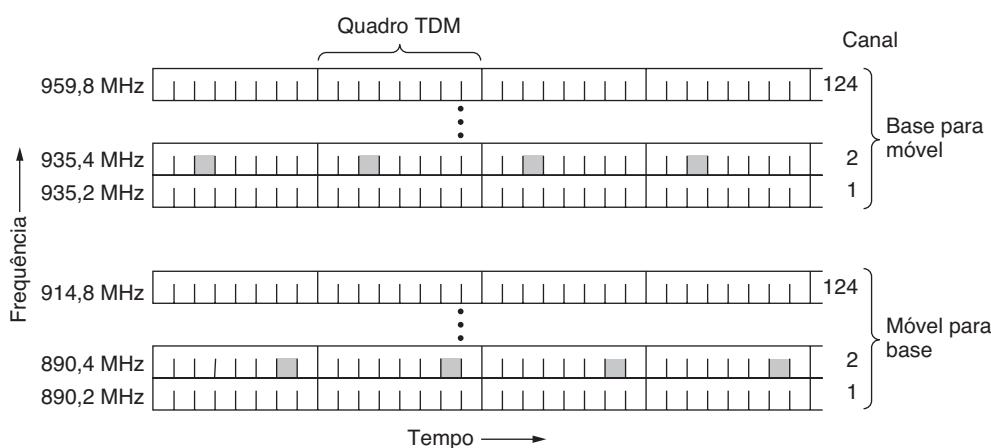



Figura 2.41 | O sistema GSM utiliza 124 canais de frequência, cada um usando um sistema TDM de oito slots.

64 kbps do PCM para sinais de voz não compactados na rede de telefonia fixa, a compactação no dispositivo móvel pode alcançar esses níveis com pouca perda de qualidade.

Como podemos ver na Figura 2.42, oito quadros de dados formam um quadro TDM, e 26 quadros TDM formam um multiquadro de 120 ms. Dos 26 quadros TDM em um multiquadro, o slot 12 é usado para controle e o slot 25 é reservado para uso futuro; assim, somente 24 estão disponíveis para tráfego do usuário.

Porém, além do multiquadro de 26 slots mostrado na Figura 2.42, também é usado um multiquadro de 51 slots (não mostrado). Alguns desses slots são empregados para guardar diversos canais de controle usados para gerenciar o sistema. O **canal de controle de broadcast** é um fluxo contínuo de saída da estação-base, contendo a identidade da estação-base e o status do canal. Todas as estações móveis monitoram a intensidade de seu sinal para verificar quando elas são transferidas para uma nova célula.

O **canal de controle dedicado** é usado para atualização de local, registro e estabelecimento de chamadas. Em particular, cada estação-base mantém o VLR, um banco de dados das estações móveis que atualmente estão sob sua jurisdição. As informações necessárias para manter o VLR são enviadas no canal de controle dedicado.

Por fim, existe o **canal de controle comum**, dividido em três subcanais lógicos. O primeiro deles é o **canal de localização**, que a estação-base utiliza para anunciar as chamadas recebidas. Cada estação móvel monitora continuamente esse canal para verificar se há chamadas a que ela deva responder. O segundo é o **canal de acesso aleatório**, que permite aos usuários solicitarem um slot no canal de controle dedicado. Se duas solicitações colidirem, elas serão adulteradas e terão de ser repetidas mais tarde. Usando o slot do canal de controle dedicado, a estação pode estabelecer uma chamada. O slot atribuído é anunciado no terceiro subcanal, o **canal de concessão de acesso**.

Por último, o GSM difere do AMPS no modo como o handoff é tratado. No AMPS, o MSC o controla totalmente, sem ajuda dos dispositivos móveis. Com os slots de tempo no GSM, na maior parte das vezes o dispositivo móvel não está nem enviando nem recebendo. Os slots ociosos são uma oportunidade para o dispositivo móvel medir a qualidade do sinal até outras estações-base nas proximidades. Ele faz isso e envia essa informação ao BSC. Este pode utilizá-la para determinar quando um dispositivo móvel está saindo de uma célula e entrando em outra, de modo que possa realizar o handoff. Esse projeto é conhecido como **handoff auxiliado pela unidade móvel**, ou **MAHO** (**Mobile Assisted HandOff**).

2.7.3 Telefones móveis de terceira geração (3G): voz e dados digitais

A primeira geração de telefones móveis foi a voz analógica, e a segunda foi a voz digital. A terceira geração de telefones móveis, ou **3G**, como é chamada, trata de voz *e* dados digitais.

Diversos fatores estão direcionando o setor. Primeiro, o tráfego de dados já ultrapassa o tráfego de voz na rede fixa e está crescendo exponencialmente, enquanto o tráfego de voz está basicamente estabilizado. Muitos especialistas da indústria esperam que o tráfego de dados também domine a voz nos dispositivos móveis, e em pouco tempo. Segundo, os setores de telefonia, entretenimento e informática passaram para a tecnologia digital e estão convergindo rapida-

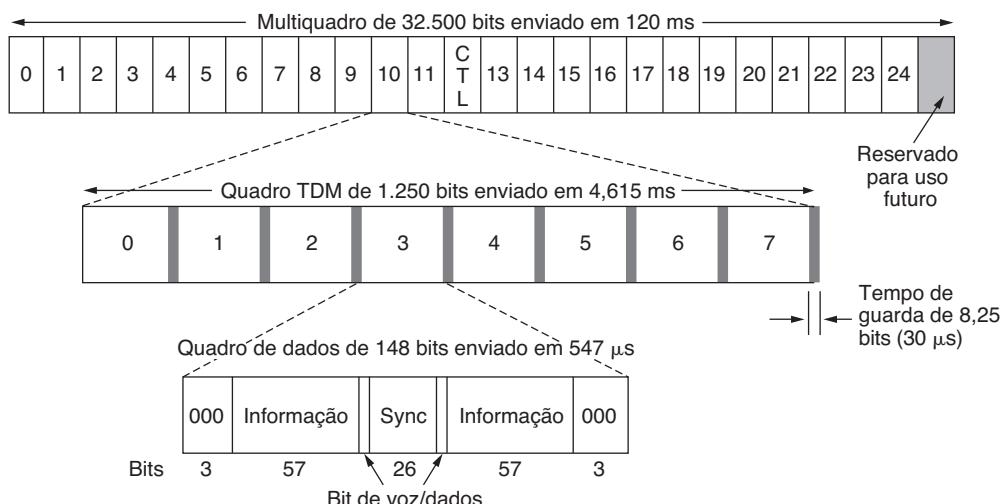



Figura 2.42 | Uma parte da estrutura de enquadramento GSM.

mente. Muitas pessoas estão utilizando dispositivos leves, portáteis, que atuam como telefone, aparelho de música e vídeo, terminal de correio eletrônico, interface Web, máquina de jogos e outros, tudo com conectividade mundial sem fios à Internet em alta largura de banda.

O iPhone da Apple é um bom exemplo desse tipo de dispositivo 3G. Com ele, as pessoas são conectadas a serviços de dados sem fio, e os volumes de dados sem fio da AT&T estão aumentando rapidamente com a popularidade dos iPhones. O problema é que o iPhone usa uma rede 2,5G (uma rede 2G melhorada, mas não uma rede 3G verdadeira), e não existe capacidade de dados suficiente para manter os usuários satisfeitos. A telefonia móvel 3G trata de fornecer largura de banda sem fio suficiente para manter esses futuros usuários satisfeitos.

Em 1992, a ITU tentou ser um pouco mais específica em relação a esse sonho e apresentou um projeto para alcançá-lo, denominado **IMT-2000**, em que IMT significava **International Mobile Telecommunications** (**telecomunicações móveis internacionais**). Os serviços básicos que a rede IMT-2000 deveria oferecer a seus usuários eram:

1. Transmissão de voz de alta qualidade.
2. Serviço de mensagens (substituindo correio eletrônico, fax, SMS, bate-papo etc.).
3. Multimídia (reprodução de música, exibição de vídeos, filmes, televisão etc.).
4. Acesso à Internet (navegação na Web, incluindo páginas com áudio e vídeo).

Outros serviços poderiam ser: videoconferência, telepresença, jogos em grupo e m-commerce (comércio móvel, bastando utilizar seu telefone no caixa para pagar as compras feitas em uma loja). Além disso, todos esses serviços deveriam estar disponíveis em âmbito mundial (com conexão automática via satélite, quando não for possível localizar nenhuma rede terrestre), de forma instantânea (sempre ativos) e com garantias de qualidade de serviço.

A ITU previu uma única tecnologia mundial para o IMT-2000, de forma que os fabricantes fossem capazes de construir um único dispositivo que pudesse ser vendido e utilizado em qualquer lugar do mundo (como reprodutores de CDs e computadores, mas diferente de telefones celulares e televisores). Ter uma única tecnologia também facilitaria bastante a vida dos operadores de redes e encorajaria mais pessoas a usarem os serviços. Guerras de formatos, como a que ocorreu entre o Betamax e o VHS quando surgiram os primeiros gravadores de vídeo, não são boas para o comércio.

Acontece que isso foi um pouco otimista. O número 2000 significou três coisas: (1) o ano em que o serviço deveria ser iniciado; (2) a frequência em que ele deveria operar (em MHz); e (3) a largura de banda que o serviço deveria ter (em kbps). Nenhum desses três se concretizou. Nada foi implementado em 2000. A ITU recomendou que todos os governos reservassem o espectro em 2 GHz para que os dispositivos pudessem passar de um país para outro de forma transparente. A China reservou a largura de banda exigida, mas ninguém mais fez isso. Finalmente, reconheceu-se que 2 Mbps não são atualmente viáveis para usuários que se movimentam *muito* (em virtude da dificuldade de realizar handoffs com rapidez suficiente). O mais realista é 2 Mbps para usuários que não estão em movimento (o que competirá de igual para igual com o ADSL), 384 kbps para pessoas andando e 144 kbps para conexões em carros.

Apesar desses problemas iniciais, muita coisa foi feita desde então. Várias propostas do IMT foram feitas e, após uma seleção, elas se reduziram a duas principais. A primeira, o **CDMA de banda larga**, ou **WCDMA** (**Wideband CDMA**), foi proposta pela Ericsson e adotada pela União Europeia, que o chamou de **sistema universal de telecomunicações móveis**, ou **UMTS** (**Universal Mobile Telecommunications System**). O outro concorrente era o **CDMA2000**, proposto pela Qualcomm.

Esses dois sistemas são mais semelhantes do que diferentes, pois são baseados no CDMA de banda larga; o WCDMA usa canais de 5 MHz e o CDMA2000 usa canais de 1,25 MHz. Se os engenheiros da Ericsson e da Qualcomm fossem confinados em uma sala e solicitados a apresentar um projeto comum, eles provavelmente conseguiriam fazê-lo rapidamente. A dificuldade é que o problema real não é de engenharia, mas político (como sempre). A Europa queria um sistema que trabalhasse junto com o GSM, enquanto os Estados Unidos queriam um sistema que fosse compatível com um sistema já amplamente desenvolvido nos Estados Unidos (o IS-95). Cada lado também apoiava sua empresa local (a Ericsson está sediada na Suécia; a Qualcomm, na Califórnia). Por fim, a Ericsson e a Qualcomm estavam envolvidas em numerosos processos relacionados a suas respectivas patentes de CDMA.

No mundo inteiro, 10 a 15 por cento dos assinantes móveis já usam tecnologias 3G. Na América do Norte e na Europa, cerca de um terço dos assinantes móveis são 3G. O Japão foi um dos primeiros a adotá-lo, e agora quase todos os telefones móveis no Japão são 3G. Esses valores incluem a implementação de UMTS e CDMA2000, e o 3G continua a ser uma grande caldeirão de atividade à medida que o mercado se aquece. Para aumentar a confusão, o UMTS se tornou um padrão 3G único com múltiplas opções incompatíveis, incluindo o CDMA2000. Essa mudança foi um esforço para unificar os vários campos, mas ele apenas encobre as diferenças técnicas e oculta o foco dos esforços em andamento. Usaremos o UMTS para indicar o WCDMA, uma forma diferente com origem no CDMA2000.

Vamos voltar nossa discussão para o uso do CDMA em redes celulares, por ser o fator de distinção dos dois sistemas. O CDMA não é FDM nem TDM, mas um tipo de mistura em que cada usuário envia na mesma banda de frequência ao mesmo tempo. Quando ele foi proposto inicialmente para sistemas de celular, a indústria teve apro-

ximadamente a mesma reação que Colombo provocou na Rainha Isabel quando ele propôs alcançar as Índias navegando na direção errada. Porém, pela persistência de uma única empresa, a Qualcomm, o CDMA teve sucesso como um sistema 2G (IS-95) e amadureceu ao ponto de se tornar a base técnica para o 3G.

Para fazer o CDMA funcionar no ambiente de telefonia móvel, é preciso mais do que a técnica de CDMA básica que descrevemos na seção anterior. Especificamente, descrevemos o CDMA síncrono, no qual as sequências de chip são exatamente ortogonais. Esse projeto funciona quando todos os usuários estão sincronizados no momento inicial de suas sequências de chip, como no caso da estação-base transmitindo para unidades móveis. A estação-base pode transmitir as sequências de chip começando ao mesmo tempo, de modo que os sinais sejam ortogonais e capazes de ser separados. Contudo, é difícil sincronizar as transmissões de telefones móveis independentes. Sem cuidado, suas transmissões chegariam na estação-base em momentos diferentes, sem nenhuma garantia de ortogonalidade. Para que as unidades móveis enviem para a estação-base sem sincronização, queremos codificar sequências que são ortogonais umas às outras em todos os deslocamentos possíveis, não apenas quando elas estão alinhadas no início.

Embora não seja possível encontrar sequências que sejam exatamente ortogonais para esse caso geral, longas sequências pseudoaleatórias chegam perto o suficiente. Elas têm a propriedade de que, com alta probabilidade, tenham uma baixa **correlação cruzada** entre si em todos os deslocamentos. Isso significa que, quando uma sequência é multiplicada por outra e somada para calcular o produto interno, o resultado será pequeno; ele seria zero se todas fossem ortogonais. (Intuitivamente, as sequências aleatórias sempre deverão parecer diferentes uma da outra. Multiplicá-las deverá, então, produzir um sinal aleatório, que se somará a um resultado pequeno.) Isso permite que um receptor filtre transmissões indesejadas do sinal recebido. Além disso, a **autocorrelação** de sequências pseudoaleatórias também é pequena, com alta probabilidade, exceto em um deslocamento zero. Isso significa que, quando uma sequência é multiplicada por uma cópia atrasada de si mesma e somada, o resultado será pequeno, exceto quando o atraso é zero. (Intuitivamente, uma sequência aleatória atrasada se parece com uma sequência aleatória diferente, e voltamos ao caso da correlação cruzada.) Isso permite que um receptor intercepte o início da transmissão desejada no sinal recebido.

O uso de sequências pseudoaleatórias permite que a estação-base receba mensagens CDMA de unidades móveis não sincronizadas. Contudo, uma suposição implícita em nossa discussão do CDMA é que os níveis de potência de todas as unidades móveis são iguais no receptor. Se não forem, uma pequena correlação cruzada com um sinal poderoso poderia superar uma grande autocorrelação com um sinal fraco. Assim, a potência de transmissão nas unidades móveis deve ser controlada para reduzir a interferência entre sinais concorrentes. É essa interferência que limita a capacidade de sistemas CDMA.

Os níveis de potência recebidos em uma estação-base dependem da distância em que os transmissores se encontram e também de quanta potência eles transmitem. Pode haver muitas estações móveis em distâncias variadas da estação-base. Uma boa heurística para equalizar a potência recebida é que cada estação móvel transmita para a estação-base no inverso do nível de potência que ela recebe dessa estação-base. Em outras palavras, uma estação móvel recebendo um sinal fraco da estação-base usará mais potência do que outra obtendo um sinal forte. Para aumentar a precisão, a estação-base também oferece feedback a cada unidade móvel para aumentar, diminuir ou manter constante sua potência de transmissão. O feedback é frequente (1.500 vezes por segundo), pois o bom controle de potência é importante para reduzir a interferência.

Outra melhoria em relação ao esquema CDMA básico que descrevemos anteriormente é permitir que diferentes usuários enviem dados em diferentes taxas. Esse truque é realizado naturalmente no CDMA fixando a taxa em que os chips são transmitidos e atribuindo aos usuários sequências de chips de diferentes tamanhos. Por exemplo, no WCDMA, a taxa de chip é de 3,84 Mchips/s, e o espalhamento de códigos varia de 4 a 256 chips. Com um código de 256 chips, restam cerca de 12 kbps após a correção de erro, e essa capacidade é suficiente para uma chamada de voz. Com um código de 4 chips, a taxa de dados do usuário é próxima de 1 Mbps. Códigos de tamanho intermediário geram taxas intermediárias; para conseguir múltiplos Mbps, a unidade móvel precisa usar mais de um canal de 5 MHz ao mesmo tempo.

Agora, vamos descrever as vantagens do CDMA, depois que explicamos os problemas para fazê-lo funcionar. Ele tem três vantagens principais. Primeiro, esse sistema pode melhorar a capacidade, tirando proveito de pequenos períodos quando alguns transmissores estão silenciosos. Nas chamadas de voz, em conversas educadas, uma parte fica em silêncio enquanto a outra fala. Em média, a linha está ocupada apenas 40 por cento do tempo. Contudo, as pausas podem ser pequenas e são difíceis de prever. Com sistemas TDM ou FDM, não é possível reatribuir slots de tempo ou canais de frequência com rapidez suficiente para se beneficiar desses pequenos silêncios. Contudo, no CDMA, um usuário reduz a interferência para outros usuários simplesmente não transmitindo, e é provável que alguma fração dos usuários não esteja transmitindo em uma célula ocupada em determinado momento. Assim, o CDMA tira proveito dos silêncios esperados para permitir um número maior de chamadas simultâneas.

Em segundo lugar, com o CDMA, cada célula usa as mesmas frequências. Diferente de GSM e AMPS, a FDM não é necessária para separar as transmissões de diferentes usuários. Isso elimina complicadas tarefas de planejamento de

frequência e melhora a capacidade. Isso também torna mais fácil para a estação-base usar várias antenas direcionais, ou **antenas setorizadas**, em vez de uma antena omnidirecional. As antenas direcionais concentram um sinal na direção desejada e o reduzem, reduzindo, portanto, a interferência em outras direções. Isso, por sua vez, aumenta a capacidade. Três projetos de setor são comuns. A estação-base precisa rastrear a unidade móvel enquanto ela se move de um setor para outro. Esse rastreamento é fácil com o CDMA, pois todas as frequências são usadas em todos os setores.

Em terceiro lugar, o CDMA facilita o **soft handoff**, em que a unidade móvel é aceita pela nova estação-base antes de a anterior se desconectar. Desse modo, não existe perda de continuidade. O soft handoff aparece na Figura 2.43. Ele é fácil com o CDMA, pois todas as frequências são usadas em cada célula. Como opção há um **hard handoff**, em que a estação-base antiga libera a chamada antes de ela ser aceita pela nova. Se a nova não for capaz de aceitá-la (por exemplo, porque não existe nenhuma frequência disponível), a chamada será desconectada de forma brusca. Os usuários tendem a notar essa interrupção, mas ela ocasionalmente é inevitável com a estrutura atual. O hard handoff é um padrão em FDM para evitar o custo de fazer a unidade móvel transmitir ou receber em duas frequências simultaneamente.

Muito se tem escrito sobre o 3G, na maior parte elogiando-o como a maior invenção desde o pão fatiado. Enquanto isso, muitas operadoras estão dando um passo cauteloso em direção a essa tecnologia, chegando ao que se costuma chamar às vezes de **2,5G**, embora a identificação 2,1G talvez seja mais precisa. Um sistema desse tipo utiliza **taxas de dados aperfeiçoadas para evolução do GSM**, ou **EDGE** (**Enhanced Data rates for GSM Evolution**), que é simplesmente o GSM com mais bits por símbolo. O problema é que mais bits por símbolo também significa mais erros por símbolo, e assim o EDGE tem nove esquemas diferentes para modulação e correção de erros, que se distinguem pela proporção da largura de banda dedicada à correção dos erros introduzidos pela velocidade mais alta. EDGE é um passo na direção a um caminho evolutivo que é definido do GSM ao WCDMA. De modo semelhante, existe um caminho evolutivo definido para as operadoras migrarem de redes IS-95 para CDMA2000.

Embora as redes 3G ainda não estejam plenamente desenvolvidas, alguns pesquisadores consideram essa tecnologia como um trabalho encerrado. Essas pessoas já estão trabalhando em sistemas 4G sob o nome de **evolução de longo prazo**, ou **LTE** (**Long Term Evolution**). Alguns dos recursos propostos por sistemas 4G incluem: alta largura de banda; ubiquidade (conectividade em toda parte); integração transparente com outras redes IP com e sem fios, incluindo pontos de acesso 802.11; gerenciamento adaptativo de recurso e espectro; e alta qualidade de serviço para multimídia. Para obter mais informações, consulte Astely et al. (2009) e Larmo et al. (2009).

Enquanto isso, redes sem fio com níveis de desempenho 4G já estão disponíveis. O principal exemplo é o **802.16**, também conhecido como **WiMAX**. Para obter uma visão geral do WiMAX móvel, consulte Ahmadi (2009). Dizer que a indústria se encontra em um estado de enorme convergência é uma grande simplificação. Espere mais alguns anos para ver o que acontece.

2.8 Televisão a cabo

Neste ponto, já estudamos os sistemas de telefonia fixa e sem fios, com uma quantidade razoável de detalhes. Sem dúvida, ambos desempenharão um papel fundamental nas futuras redes. Porém, há uma outro participante importante que surgiu na última década para se obter acesso à Internet: as redes de televisão a cabo. Muitas pessoas já recebem seu serviço telefônico e de Internet por cabo. Nas próximas seções, examinaremos com mais detalhes a televisão a cabo como uma rede e vamos compará-la com os sistemas de telefonia que acabamos de estudar. Para obter mais informações sobre sistemas a cabo, consulte Donaldson e Jones (2001), Dutta-Roy (2001) e Fellows e Jones (2001).

2.8.1 CATV (Community Antenna Television)

A televisão a cabo foi concebida no final da década de 1940 como uma forma de proporcionar melhor recepção às pessoas que vivem em áreas rurais ou montanhosas. No início, o sistema consistia em uma grande antena situada no alto de uma colina para captar o sinal de televisão que se propaga pelo ar, um amplificador chamado **headend**,

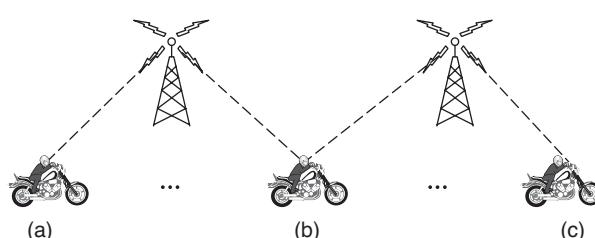



Figura 2.43 | Soft handoff (a) antes, (b) durante e (c) depois.

para reforçá-lo, e um cabo coaxial para distribuí-lo pelas casas das pessoas, como ilustra a Figura 2.44.

Nos primeiros anos, a TV a cabo era chamada **televisão de antena comunitária**, ou **CATV** (**Community Antenna Television**). Sua operação era muito simples; qualquer pessoa que tivesse alguma prática em eletrônica era capaz de instalar um serviço para sua cidade, e os usuários se reuniam para pagar os custos do serviço. À medida que o número de assinantes crescia, outros cabos eram conectados ao cabo original e eram acrescentados outros amplificadores conforme a necessidade. A transmissão era unidirecional, do headend para os usuários. Em 1970, havia milhares de sistemas independentes.

Em 1974, a Time Inc. lançou um novo canal, denominado Home Box Office, com novo conteúdo (filmes) e distribuído somente por cabo. Seguiram-se outros canais dedicados apenas a notícias, esportes, culinária e muitos outros temas. Esse desenvolvimento ocasionou duas mudanças na indústria. Primeiro, as grandes corporações começaram a adquirir os sistemas a cabo existentes e estender novos cabos para conquistar novos assinantes. Segundo, agora havia necessidade de conectar vários sistemas, normalmente em cidades distantes, a fim de distribuir o conteúdo dos novos canais de TV a cabo. As empresas de TV a cabo começaram a estender cabos entre as cidades para conectar todas elas em um único sistema. Esse padrão era semelhante ao que aconteceu na indústria de telefonia oitenta anos antes, com a conexão de estações finais anteriormente isoladas para tornar possível a comunicação interurbana.

2.8.2 Internet por cabo

Com o passar dos anos, o sistema de TV a cabo cresceu, e os cabos entre as várias cidades foram substituídos por fibra óptica de alta largura de banda, de forma semelhante ao que aconteceu no sistema telefônico. Um sistema com fibra nas linhas principais e cabo coaxial nas ligações para as residências é chamado **sistema híbrido de cabo coaxial e fibra**, ou **HFC** (**Hybrid Fiber Coax**). Os conversores eletro-ópticos que constituem a interface entre as partes óptica e elétrica do sistema são chamados **nós de fibra**. Pelo fato de a largura de banda da fibra ser muito maior que a dos cabos coaxiais, um nó de fibra pode alimentar vários cabos coaxiais. A Figura 2.45(a) mostra uma parte de um sistema HFC moderno.

Durante a última década, muitas operadoras de TV a cabo decidiram entrar no ramo de acesso à Internet e, muitas vezes, também no ramo de telefonia. No entanto, diferenças técnicas entre as instalações de cabo e de telefonia têm efeito sobre o que deve ser realizado para alcançar esses objetivos. Por um lado, todos os amplificadores unidirecionais no sistema tinham de ser substituídos por amplificadores bidirecionais, para dar suporte a transmissões tanto upstream quanto downstream. Enquanto isso estava acontecendo, os primeiros sistemas de Internet por cabo usavam a rede de televisão para as transmissões downstream e uma conexão discada via rede telefônica para as transmissões upstream. Essa foi uma alternativa esperta, mas não de rede, em comparação com o que poderia ser.

No entanto, há outra diferença entre o sistema HFC da Figura 2.45(a) e o sistema telefônico da Figura 2.45(b), a qual é muito mais difícil de remover. Nos bairros, um único cabo é compartilhado por muitas casas, ao passo que, no sistema telefônico, cada casa tem seu próprio circuito terminal privado. Quando é utilizado para difusão de televisão, esse compartilhamento é natural. Todos os programas são transmitidos no cabo e não importa se existem 10 ou 10.000 espectadores. No entento, quando o mesmo cabo é usado para acesso à Internet, faz uma grande diferença a existência de 10 ou de 10.000 usuários. Se um usuário decidir baixar um arquivo muito grande, essa largura de banda estará potencialmente sendo retirada de outros usuários. Quanto mais usuários, maior a competição pela largura de banda. O sistema de telefonia não tem essa propriedade específica: a transferência de um grande arquivo por uma linha ADSL não reduz a largura de banda de seu vizinho. Por outro lado, a largura de banda do cabo coaxial é muito mais alta do que a dos pares trançados, então você terá sorte se seus vizinhos não usarem muito a Internet.

A estratégia usada pela indústria de serviços a cabo para resolver esse problema é desmembrar cabos longos e conectar cada um deles diretamente a um nó de fibra. A largura de banda do headend até cada nó de fibra é efetivamente infi-

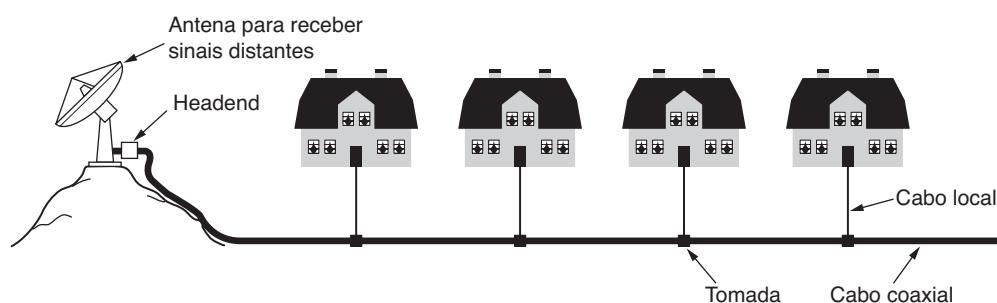



Figura 2.44 | Um antigo sistema de TV a cabo.

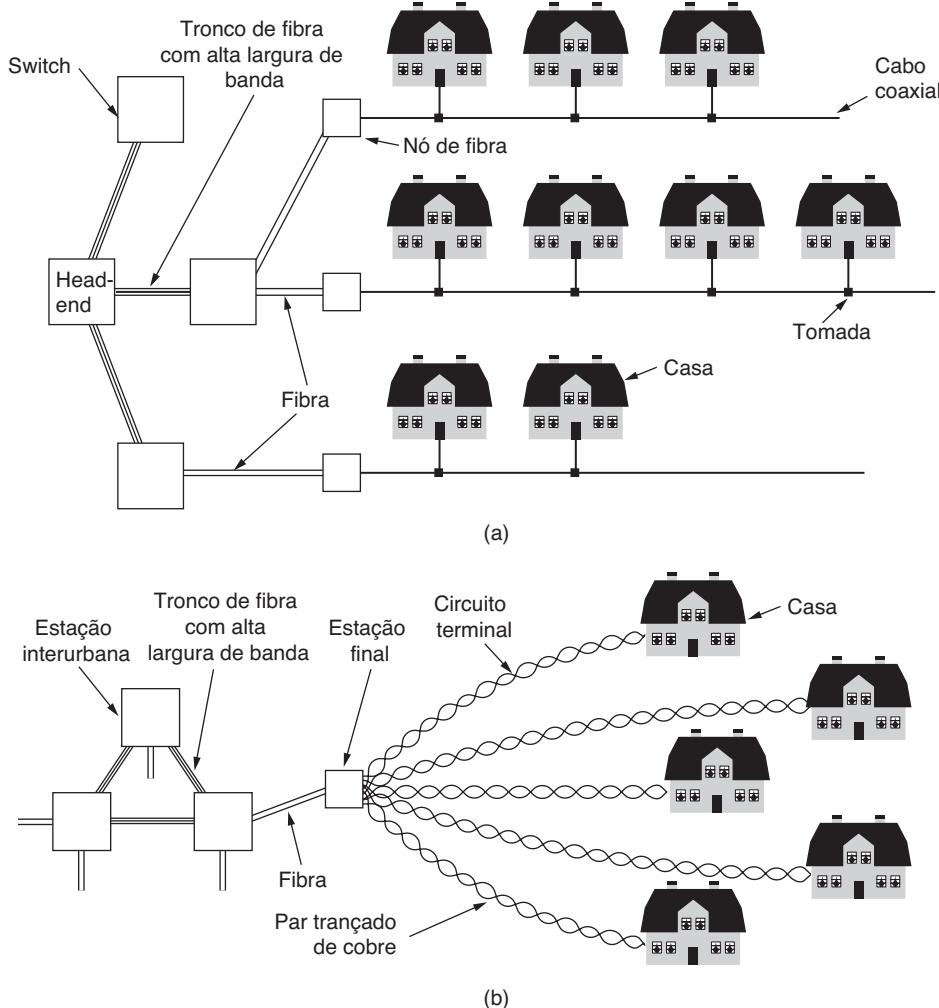



Figura 2.45 | (a) Televisão a cabo. (b) O sistema de telefonia fixa.

nita e, se não existirem muitos assinantes em cada segmento de cabo, o volume de tráfego será gerenciável. Os cabos típicos atuais conectam de 500 a 2.000 casas; porém, à medida que um número cada vez maior de pessoas adquire o serviço de Internet por cabo, a carga pode se tornar grande demais, exigindo mais divisões e mais nós de fibra.

2.8.3 Alocação do espectro

Eliminar todos os canais de TV e usar a infraestrutura de cabo estritamente para acesso à Internet provavelmente geraria um número razoável de clientes insatisfeitos; assim, as empresas de TV a cabo hesitam em fazê-lo. Além disso, a maioria das cidades tem uma regulamentação bastante pesada sobre o que é transmitido por cabo e, portanto, as operadoras de serviços não teriam permissão para fazer isso, ainda que desejassem. Como consequência, elas precisaram encontrar um modo de fazer a televisão e a Internet coexistirem no mesmo cabo.

A solução é contar com a multiplexação por divisão de frequência. Os canais de TV a cabo da América do Norte ocupam a região de 54 a 550 MHz (com exceção do rádio FM, que ocupa a faixa de 88 a 108 MHz). Esses canais têm 6 MHz de largura, incluindo as bandas de proteção, e podem transportar um canal de TV analógica tradicional ou vários canais de TV digital. Na Europa, a extremidade inferior em geral é de 65 MHz, e os canais têm de 6 a 8 MHz de largura, em virtude da maior resolução exigida pelos sistemas PAL e SECAM, mas o esquema de alocação é semelhante nos outros aspectos. A parte baixa da banda não é usada. Os cabos modernos também operam bem acima de 550 MHz, chegando frequentemente a 750 MHz ou mais. A solução escolhida foi introduzir upstream na banda de 5 a 42 MHz (um pouco mais alta na Europa) e usar as frequências na extremidade alta para o fluxo downstream. O espectro dos serviços de cabo é ilustrado na Figura 2.46.

Observe que, como todos os sinais de televisão são downstream, é possível usar amplificadores upstream que

só funcionam na região de 5 a 42 MHz e amplificadores downstream que só funcionam na frequência de 54 MHz e acima desta, como mostra a figura. Desse modo, obtemos uma assimetria nas larguras de banda upstream e downstream, porque está disponível uma parte maior do espectro acima da faixa de TV do que abaixo dela. Por outro lado, a maior parte do tráfego provavelmente será downstream, e, assim, as operadoras de serviços a cabo não estão insatisfeitas com esse fato. Como vimos antes, em geral as companhias telefônicas oferecem um serviço DSL assimétrico, embora não tenham nenhuma razão técnica para fazê-lo.

Além de atualizar os amplificadores, a operadora também tem de atualizar o headend, que deve passar de um amplificador não inteligente para um sistema computadorizado digital inteligente, com uma interface de fibra de alta largura de banda para um ISP. Frequentemente, o nome também é atualizado, passando de 'headend' para **sistema de terminação de modem a cabo**, ou **CMTS** (**Cable Modem Termination System**). No texto a seguir, evitaremos realizar uma atualização de nome e continuaremos usando o tradicional termo 'headend'.

2.8.4 Modems a cabo

O acesso à Internet exige um modem a cabo, um dispositivo que tem duas interfaces: uma para o computador e outra para a rede a cabo. Nos primeiros anos da Internet a cabo, cada operadora tinha um modem a cabo próprio, patenteado, instalado por um técnico da empresa de serviços a cabo. Entretanto, logo se tornou óbvio que um padrão aberto criaria um mercado competitivo de modems a cabo e reduziria os preços, encorajando, assim, o uso do serviço. Além disso, fazer os clientes comprarem e instalarem eles próprios os modems a cabo (como fazem com pontos de acesso wireless) eliminaria os temidos problemas de assistência técnica.

Consequentemente, as maiores operadoras de serviços a cabo se juntaram a uma empresa chamada CableLabs para produzir um padrão de modem a cabo e testar a compatibilidade dos produtos. Esse padrão, chamado **especificação da interface de serviço de dados por cabo**, ou **DOCSIS** (**Data Over Cable Service Interface Specification**), substituiu a maioria dos modems patenteados. O padrão DOCSIS versão 1.0 apareceu em 1997, e logo foi seguido pelo DOCSIS 2.0, em 2001. Ele aumentava as taxas upstream para dar melhor suporte a serviços simétricos, como a telefonia IP. A versão mais recente do padrão é DOCSIS 3.0, que apareceu em 2006. Ele usa mais largura de banda para aumentar as taxas nas duas direções. A versão europeia desses padrões é chamada **EuroDOCSIS**. No entanto, nem todas as operadoras de cabo gostam da ideia de um padrão, pois muitas delas estavam ganhando muito dinheiro alugando seus modems para seus clientes cativos. Um padrão aberto, com dezenas de fabricantes vendendo modems a cabo nas lojas, encerra essa prática lucrativa.

A interface entre o modem e o computador é simples. Em geral, ela é feita por Ethernet ou, ocasionalmente, por USB. A outra extremidade é mais complicada, pois usa toda a FDM, a TDM e o CDMA para compartilhar a largura de banda do cabo entre os assinantes.

Quando um modem a cabo é conectado e ligado, ele percorre os canais downstream procurando por um pacote especial emitido periodicamente pelo headend para fornecer parâmetros de sistema aos modems que acabaram de se conectar. Ao encontrar esse pacote, o novo modem anuncia sua presença em um dos canais upstream. O headend responde atribuindo o modem a seus canais upstream e downstream. Essas atribuições podem ser alteradas mais tarde, se o headend julgar necessário equilibrar a carga.

O uso de canais de 6 MHz ou 8 MHz é a parte FDM. Cada modem a cabo envia dados em um canal upstream e em um canal downstream, ou em vários canais sob DOCSIS 3.0. O esquema normal é conseguir cada canal downstream de 6 (ou 8) MHz e modulá-lo com QAM-64 ou, se a qualidade do cabo for excepcionalmente boa, com QAM-256. Com um canal de 6 MHz e QAM-64, obtemos cerca de 36 Mbps. Quando o overhead é subtraído, a carga útil resultante é de cerca de 27 Mbps. Com QAM-256, a carga útil resultante é de cerca de 39 Mbps. Os valores europeus são 1/3 maiores.

Para upstream, há mais ruído de RF, pois o sistema não foi projetado originalmente para dados, e o ruído de vários

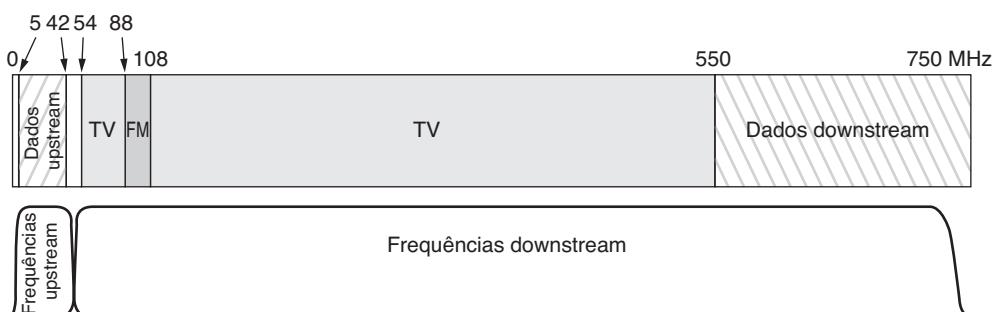



Figura 2.46 Alocação de frequências em um sistema típico de TV a cabo usado para acesso à Internet.

assinantes é afunilado no headend, de modo que é utilizado um esquema mais conservador. Este varia de QPSK a QAM-128, em que alguns dos símbolos são usados para proteção de erro com a modulação codificada por treliças. Com menos bits por símbolo no upstream, a assimetria entre as taxas upstream e downstream é muito mais do que é sugerido pela Figura 2.46.

A TDM é então usada para compartilhar largura de banda no upstream para diversos assinantes. De outra forma, suas transmissões colidiriam no headend. O tempo é dividido em **minislots**, e diferentes assinantes podem enviar em diferentes minislots. Para que isso funcione, o modem determina sua distância até o headend, enviando-lhe um pacote especial e verificando quanto tempo demora para receber a resposta. Esse processo é chamado **verificação do alcance**. É importante para o modem conhecer sua distância, a fim de se acomodar ao modo de operação dos canais upstream e obter a sincronização correta. Cada pacote upstream deve caber em um ou mais minislots consecutivos no headend quando é recebido. O headend anuncia periodicamente o início de uma nova rodada de minislots, mas o tiro de partida não é ouvido em todos os modems ao mesmo tempo, em virtude do tempo de propagação no cabo. Conhecendo a que distância está do headend, cada modem pode calcular há quanto tempo o primeiro minislot realmente começou. A extensão do minislot depende da rede. Uma carga útil típica é de 8 bytes.

Durante a inicialização, o headend também atribui cada modem a um minislot, que será usado para solicitar largura de banda upstream. Quando um computador quer enviar um pacote, ele o transfere ao modem, que então solicita o número necessário de minislots. Se a solicitação for aceita, o headend colocará uma confirmação no canal downstream, informando ao modem quais minislots foram reservados para seu pacote. Este é então enviado, a partir do minislot alocado a ele. Pacotes adicionais podem ser solicitados com a utilização de um campo no cabeçalho.

Em geral, vários modems receberão o mesmo minislot, o que leva à disputa. Há duas possibilidades diferentes para lidar com isso. A primeira é que o CDMA é usado para compartilhar o minislot entre os assinantes. Isso resolve o problema da disputa, pois todos os assinantes com sequência de código CDMA podem enviar ao mesmo tempo, embora com uma taxa reduzida. A segunda opção é que o CDMA não seja utilizado, quando não haverá confirmação, em decorrência de uma colisão. Nesse caso, o modem simplesmente esperará um tempo aleatório e tentará de novo. Após cada falha sucessiva, o tempo da aleatoriedade é dobrado. (Para os leitores que já estão um pouco familiarizados com as redes, esse algoritmo é simplesmente o modelo ALOHA adotado com a recuperação de erro por backoff. A Ethernet não pode ser usada em redes a cabo, pois as estações não conseguem detectar o meio compartilhado. Voltaremos a analisar essas questões no Capítulo 4.)

Os canais downstream são gerenciados de modo diferente dos canais upstream. Por um lado, só existe um transmissor (o headend) e, assim, não há disputa nem a necessidade de minislots, que, na realidade, são apenas a multiplexação estatística por divisão de tempo. Por outro lado, o tráfego downstream em geral é muito maior que o upstream, e então é usado um tamanho de pacote fixo, de 204 bytes. Uma parte desse código é um código de correção de erros de Reed-Solomon e algumas outras fontes de overhead, restando uma carga útil do usuário igual a 184 bytes. Esses números foram escolhidos para manter a compatibilidade com a televisão digital usando MPEG-2, de forma que os canais de TV e os canais de dados downstream sejam formatados de maneira idêntica. As conexões lógicas estão representadas na Figura 2.47.

2.8.5 Comparação entre ADSL e cabo

O que é melhor, ADSL ou cabo? Isso é como perguntar qual sistema operacional, qual idioma ou qual religião é melhor. A resposta depende da pessoa a quem você faz a pergunta. Vamos comparar alguns aspectos das tecnologias ADSL e a cabo. Ambas utilizam fibra no backbone, mas diferem nas extremidades. O sistema a cabo utiliza cabo coaxial, enquanto o ADSL utiliza par trançado. A capacidade teórica de transporte do cabo coaxial é centenas de vezes maior que a do par trançado. Porém, a capacidade total do cabo não está disponível para os usuários de dados, pois

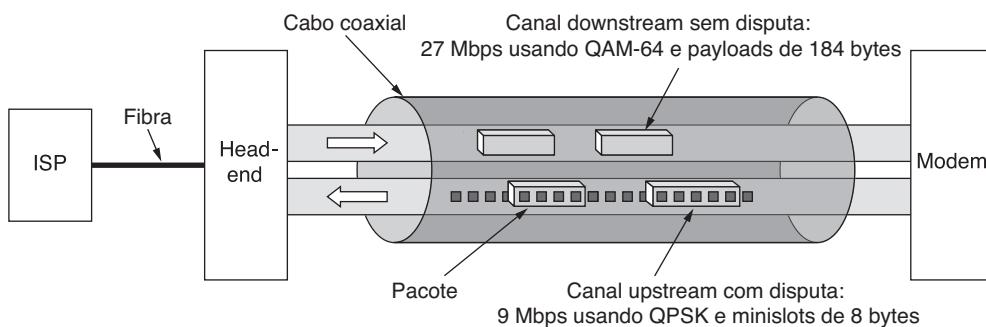



Figura 2.47 Detalhes típicos dos canais upstream e downstream na América do Norte.

grande parte da largura de banda do cabo é desperdiçada em material sem utilidade, como programas de TV.

Na prática, é difícil generalizar a respeito da capacidade efetiva. Os provedores de ADSL fazem declarações específicas sobre a largura de banda (por exemplo, 1 Mbps downstream, 256 kbps upstream) e, em geral, alcançam cerca de 80 por cento desses valores de forma consistente. Os provedores de serviços a cabo podem limitar artificialmente a largura de banda para cada usuário, ajudando-os a fazer previsões de desempenho, mas não fazem afirmação alguma, porque a capacidade efetiva depende da quantidade de pessoas atualmente ativas no segmento de cabo do usuário. Algumas vezes isso pode ser melhor que a ADSL, e outras vezes pode ser pior. Entretanto, o que talvez incomode é a imprevisibilidade. O fato de ter um ótimo serviço em um minuto não garante um ótimo serviço no minuto seguinte, pois o maior devorador de largura de banda da cidade pode ter acabado de ligar seu computador nesse momento.

À medida que um sistema ADSL conquista mais usuários, seus números crescentes têm pouco efeito sobre os usuários existentes, pois cada usuário tem uma conexão dedicada. No caso dos sistemas a cabo, quanto mais assinantes se inscrevem para receber o serviço de Internet, mais o desempenho diminui para os usuários atuais. A única solução é a operadora dos serviços a cabo dividir os cabos ocupados e conectar cada um deles diretamente a um nó de fibra. Isso custa tempo e dinheiro e, portanto, há pressões comerciais para que se evite fazê-lo.

A propósito, já estudamos outro sistema com um canal compartilhado semelhante ao cabo: o sistema de telefonia móvel. Também nesse caso, um grupo de usuários — que poderíamos chamar companheiros de célula — compartilha um volume fixo de largura de banda. Para o tráfego de voz, que é bastante suave, a largura de banda é dividida de modo rígido em blocos fixos entre os usuários ativos usando FDM e TDM. Porém, no caso do tráfego de dados, essa divisão rígida é pouco eficaz porque, com frequência, os usuários de dados estão ociosos e, nesse caso, a largura de banda reservada para eles é desperdiçada. Assim como o cabo, um meio mais dinâmico é utilizado para alocar a largura de banda compartilhada.

A disponibilidade é um ponto no qual as tecnologias ADSL e a cabo diferem. Todo mundo tem um telefone, mas nem todos os usuários estão próximos o bastante de sua estação final para receber o serviço ADSL. Por outro lado, nem todos têm a tecnologia de cabo, mas, se você tem essa tecnologia e a empresa fornece acesso à Internet, é possível obtê-lo. A distância até o nó de fibra ou até o headend não é problema. Também vale a pena notar que, como a tecnologia de cabo teve início como um meio de distribuição de televisão, poucas empresas têm esse recurso.

Sendo um meio ponto a ponto, a ADSL é inerentemente mais segura que o cabo. Qualquer usuário de serviços de cabo pode ler com facilidade todos os pacotes que passam pelo cabo. Por essa razão, qualquer provedor de serviços a cabo decente irá criptografar todo o tráfego em ambos os sentidos. Apesar disso, ainda é mais seguro impedir que seu vizinho receba absolutamente qualquer mensagem do que permitir que ele receba mensagens criptografadas destinadas a você.

Em geral, o sistema de telefonia é mais confiável do que o sistema a cabo. Por exemplo, ele tem potência de reserva e continua a funcionar normalmente, mesmo durante uma queda de energia. No caso dos sistemas a cabo, se houver falta de energia em qualquer amplificador ao longo da cadeia, todos os usuários situados abaixo dele serão instantaneamente desconectados.

Por fim, a maioria dos provedores de ADSL oferece a opção de escolher ISPs. Às vezes, eles são até mesmo obrigados a fazer isso por lei, o que nem sempre acontece no caso das operadoras de serviços a cabo.

A conclusão é que ADSL e cabo são muito mais semelhantes do que diferentes. Eles oferecem um serviço parecido e, à medida que a competição entre as duas tecnologias se aquecer, provavelmente também oferecerão preços parecidos.

2.9 Resumo

A camada física é a base de todas as redes. A natureza impõe dois limites fundamentais sobre todos os canais, e estes determinam sua largura de banda. Esses limites são o limite de Nyquist, que trata de canais sem ruídos, e o limite de Shannon, que trata de canais com ruídos.

Os meios de transmissão podem ser guiados ou não guiados. Os principais meios guiados são o par trançado, o cabo coaxial e a fibra óptica. Dentre os meios não guiados estão o rádio, as micro-ondas, os raios infravermelhos, os raios laser através do ar e os satélites.

Métodos de modulação digitais enviam bits por meios guiados e não guiados como sinais analógicos. Os códigos de linha operam na banda base, e os sinais podem ser colocados em uma banda passante por meio da modulação da amplitude, da frequência e da fase de uma portadora. Os canais podem ser compartilhados entre usuários com a multiplexação por divisão de tempo, frequência e código.

Um elemento-chave na maioria das redes a longa distância é o sistema de telefonia implantado. Seus principais componentes são os circuitos terminais, troncos e switches. A ADSL oferece velocidades de até 40 Mbps ao circuito terminal do assinante, dividindo-o em muitas subportadoras que trabalham em paralelo. Isso ultrapassa muito as taxas dos modems de telefone. PONs levam a fibra até a residência, tornando as taxas de acesso ainda maiores do que a ADSL.

Os troncos transportam informações digitais e são multiplexados com WDM para oferecer muitos enlaces de alta capacidade pelas fibras individuais, bem como TDM

para compartilhar cada enlace com taxa alta entre os usuários. A comutação de circuitos e a comutação de pacotes são importantes.

Para aplicações móveis, o sistema telefônico fixo não é adequado. Hoje, os telefones celulares estão sendo amplamente utilizados para voz, e cada vez mais são empregados na transmissão de dados. Eles passaram por três gerações. A primeira, 1G, era analógica, dominada pelo AMPS. A 2G era digital, com GSM sendo atualmente o sistema de telefonia móvel mais utilizado no mundo. A 3G é digital e se baseia no CDMA de banda larga, com WCDMA e CDMA2000 também sendo implementados agora.

Um sistema alternativo para acesso de rede é o sistema de televisão a cabo, que evoluiu gradualmente de uma antena comunitária até chegar ao sistema híbrido de cabo coaxial e fibra, e de televisão para televisão e Internet. Potencialmente, ele oferece largura de banda muito alta, mas a largura de banda disponível na prática depende muito dos outros usuários, pois ela é compartilhada.

Problemas

1. Calcule os coeficientes de Fourier para a função $f(t) = t$ ($0 \leq t \leq 1$).
2. Um canal sem ruído de 4 kHz tem uma amostra a cada 1 ms. Qual é a taxa máxima de dados desse canal? Como a taxa máxima de dados muda se o canal tiver ruído, com uma relação sinal-ruído de 30 dB?
3. Os canais de televisão têm 6 MHz. Quantos bits/s poderão ser enviados, se forem usados sinais digitais de quatro níveis? Suponha um canal sem ruído.
4. Se um sinal binário for enviado sobre um canal de 3 kHz cuja relação sinal-ruído é de 20 dB, qual será a taxa máxima de dados que poderá ser alcançada?
5. Qual é a relação sinal-ruído necessária para colocar uma portadora T1 em uma linha de 50 kHz?
6. Quais são as vantagens da fibra óptica em relação ao cobre como meio de transmissão? Existe alguma desvantagem no uso da fibra óptica em relação ao cobre?
7. Qual é a largura de banda existente em 0,1 mícron de espectro em um comprimento de onda de 1 mícron?
8. Queremos enviar uma sequência de imagens de tela de computador por uma fibra óptica. A tela tem 2.560 × 1.600 pixels, e cada pixel tem 24 bits. Há sessenta imagens de tela por segundo. Qual é a largura de banda necessária, e quantos micra de comprimento de onda são necessários para essa banda a 1,30 mícron?
9. O teorema de Nyquist também se aplica à fibra óptica de alta qualidade em modo único, ou somente ao fio de cobre?
10. Em geral, as antenas de rádio funcionam melhor quando o diâmetro da antena é igual ao comprimento das ondas de rádio. Uma variação razoável para o diâmetro das antenas é de 1 cm a 5 m. Que faixa de frequências é coberta por esse intervalo?
11. Um feixe de raios laser de 1 mm está orientado para um detector localizado a 100 m de distância do telhado de um edifício. Quanto desvio angular (em graus) o laser precisa ter antes de perder o detector?
12. Os 66 satélites de baixa órbita do projeto Iridium estão divididos em seis eixos em torno da Terra. Na altitude em que eles se encontram, o período é de noventa minutos. Qual é o intervalo médio entre handoffs no caso de um transmissor estacionário?
13. Calcule o tempo de trânsito de ponta a ponta para um pacote trafegar pelos satélites GEO (altitude: 35.800 km), MEO (altitude: 18.000 km) e LEO (altitude: 750 km).
14. Qual é a latência de uma chamada originada no Polo Norte para alcançar o Polo Sul se a chamada for roteada por satélites Iridium? Suponha que o tempo de comutação nos satélites seja 10 microssegundos e o raio da Terra seja 6.371 km.
15. Qual é a largura de banda mínima necessária para alcançar uma taxa de dados de B bits/s se o sinal for transmitido usando as codificações NRZ, MLT-3 e Manchester? Justifique sua resposta.
16. Prove que, na codificação 4B/5B, uma transição de sinal ocorrerá em tempos de pelo menos quatro bits.
17. Quantos códigos de estações finais existiam antes de 1984, quando cada estação final era identificada por seu código de área de três dígitos e pelos três primeiros dígitos do número local? Os códigos de área se iniciavam com um dígito no intervalo de 2 a 9, tinham 0 ou 1 como o segundo e terminavam com qualquer dígito. Os dois primeiros dígitos de um número local sempre estavam no intervalo de 2 a 9. O terceiro dígito podia ser qualquer um.
18. Um sistema telefônico simples consiste em duas estações finais e uma única estação interurbana, à qual cada estação final está conectada por um tronco full-duplex de 1 MHz. Um telefone comum é usado para fazer quatro ligações em um dia útil de 8 horas. A duração média de cada chamada é de 6 minutos. Dez por cento das chamadas são interurbanas (ou seja, passam pela estação interurbana). Qual é o número máximo de telefones que uma estação final pode aceitar? (Suponha 4 kHz por circuito.) Explique por que a companhia telefônica pode decidir dar suporte a um número menor de telefones do que esse número máximo na estação final.
19. Uma companhia telefônica regional tem 10 milhões de assinantes. Cada um de seus telefones está conectado a uma estação central por um fio de cobre de par trançado. O comprimento médio desses pares trançados é de 10 km. Quanto vale o cobre contido nos circuitos terminais? Suponha que a seção transversal de cada fio seja um círculo com 1 mm de diâmetro, a densidade específica do cobre seja 9,0 gramas/cm^3 e que o cobre seja vendido ao preço de 6 dólares por quilograma.

20. Um oleoduto é um sistema simplex, um sistema half-duplex, um sistema full-duplex ou nenhum dos anteriores? E um rio ou uma comunicação no estilo *walkie-talkie*?
21. O custo de um microprocessador rápido diminuiu tanto que agora é possível incluir um em cada modem. De que maneira isso afeta o tratamento de erros na linha telefônica? Isso evita a necessidade de verificação/correção de erros na camada 2?
22. Um diagrama de constelação para modems semelhante ao da Figura 2.19 tem pontos de dados nas seguintes coordenadas: (1,1), (1, –1), (–1, 1) e (–1, –1). Quantos bps um modem com esses parâmetros pode alcançar a uma taxa de transmissão de 1.200 símbolos/s?
23. Qual é a taxa de bits máxima alcançável em um modem padrão V.32 se a taxa baud for 1.200 e nenhuma correção de erro for usada?
24. Quantas frequências um modem full-duplex QAM-64 utiliza?
25. Dez sinais, cada um exigindo 4.000 Hz, são multiplexados em um único canal utilizando FDM. Qual é a largura de banda mínima exigida para o canal multiplexado? Suponha que as bandas de proteção tenham 400 Hz de largura.
26. Por que o tempo de amostragem do PCM foi definido como 125 µs?
27. Qual é o percentual de overhead em uma portadora T1, ou seja, que percentagem dos 1,544 Mbps não é entregue ao usuário final? Como isso se relaciona ao percentual de overhead nas linhas OC-1 e OC-768?
28. Compare a taxa máxima de dados de um canal sem ruído de 4 kHz usando:
    (a) Codificação analógica (por exemplo, QPSK) com 2 bits por amostra.
    (b) O sistema T1 PCM.
29. Se um sistema de portadora T1 apresentar uma falha e perder o controle de onde está, ele tentará se sincronizar novamente usando o primeiro bit de cada quadro. Em média, quantos quadros terão de ser examinados para que seja feita a ressincronização com uma probabilidade de erro de 0,001?
30. Qual é a diferença, se houver, entre a parte demoduladora de um modem e a parte codificadora de um codec? (Afinal, ambos convertem sinais analógicos em sinais digitais.)
31. Os clocks da SONET têm uma taxa de tração de aproximadamente uma parte em 10^9. Quanto tempo a tração leva para igualar a largura de 1 bit? Quais são as implicações desse cálculo, se houver?
32. Quanto tempo levará para transmitir um arquivo de 1 GB de um VSAT para outro usando um hub como o mostrado na Figura 2.14? Suponha que o uplink seja de 1 Mbps, o downlink seja de 7 Mbps e a comutação de circuitos seja usada com um tempo de configuração de circuito de 1,2 s.
33. Calcule o tempo de trânsito no problema anterior se for usada a comutação de pacotes. Suponha que o tamanho do pacote seja de 64 KB, o atraso de comutação no satélite e no hub seja de 10 microssegundos e o tamanho do cabeçalho do pacote seja 32 bytes.
34. Na Tabela 2.5, a taxa de dados do usuário para OC-3 é de 148,608 Mbps. Mostre como esse número pode ser derivado dos parâmetros OC-3 da SONET. Quais serão as taxas de dados bruta, SPE e do usuário de uma linha OC-3072?
35. Para acomodar taxas de dados mais baixas que STS-1, a SONET tem um sistema de tributários virtuais (VTs). Um VT é uma carga útil parcial que pode ser inserida em um quadro STS-1 e combinada com outras cargas úteis parciais para preencher o quadro de dados. O VT1.5 utiliza três colunas, o VT2 usa quatro colunas, o VT3 usa seis colunas e o VT6 usa 12 colunas de um quadro STS-1. Qual VT pode acomodar:
    (a) Um serviço DS-1 (1,544 Mbps)?
    (b) Um serviço europeu CEPT-1 (2,048 Mbps)?
    (c) Um serviço DS-2 (6,312 Mbps)?
36. Qual é a largura de banda disponível para o usuário em uma conexão OC-12c?
37. Três redes de comutação de pacotes possuem n nós cada uma. A primeira rede tem uma topologia em estrela com um switch central, a segunda é um anel (bidirecional) e a terceira é totalmente interconectada, com um fio interligando cada nó. Quais são as opções de caminhos de transmissão em hops no melhor caso, no caso médio e no pior caso?
38. Compare o atraso no envio de uma mensagem de x bits sobre um caminho de k hops em uma rede comutada por circuitos e em uma rede (levemente carregada) comutada por pacotes. O tempo de configuração de circuitos é de s segundos, o atraso de propagação é de d segundos por hop, o tamanho do pacote é de p bits e a taxa de dados é de b bps. Sob quais condições a rede de pacotes tem um atraso mais baixo? Além disso, explique as condições sob as quais uma rede de comutação de pacotes é preferível a uma rede de comutação de circuitos.
39. Suponha que x bits de dados do usuário tenham de ser transmitidos por um caminho de k hops em uma rede comutada por pacotes como uma série de pacotes, cada um contendo p bits de dados e h bits de cabeçalho, sendo $x \gg p + h$. A taxa de bits das linhas é b bps e o atraso de propagação é desprezível. Que valor de p minimiza o atraso total?
40. Em um sistema telefônico típico com células hexagonais, é proibido reutilizar uma banda de frequências em uma célula adjacente. Se 840 frequências estão disponíveis, quantas podem ser utilizadas em uma determinada célula?
41. O layout real de células raramente é tão regular quanto o da Figura 2.39. Mesmo as formas de células individuais em geral são irregulares. Apresente uma possível razão para isso. Como essas formas irregulares afetam a atribuição de frequência de cada célula?
42. Faça uma estimativa do número de microcélulas PCS com 100 m de diâmetro que seriam necessárias para cobrir a cidade de São Francisco (120 quilômetros quadrados).

43. Às vezes, quando um usuário móvel cruza o limite de uma célula para outra, a chamada atual é encerrada de forma brusca, embora todos os transmissores e receptores estejam funcionando perfeitamente. Por quê?

44. Suponha que *A*, *B* e *C* estejam simultaneamente transmitindo bits 0, usando um sistema CDMA com as sequências de chips da Figura 2.24(a). Qual é a sequência de chips resultante?

45. Considere um modo diferente de observar a propriedade de ortogonalidade das sequências de chips do CDMA. Cada bit em um par de sequências pode coincidir ou não. Expresse a propriedade de ortogonalidade em termos de correspondências e não correspondências.

46. Um receptor CDMA recebe os seguintes chips: (–1 +1 –3 +1 –1 –3 +1 +1). Supondo as sequências de chips definidas na Figura 2.24(b), que estações transmitiram, e quais bits cada uma enviou?

47. Na Figura 2.24, existem quatro estações que podem transmitir. Suponha que mais quatro estações sejam acrescentadas. Forneça as sequências de chips dessas estações.

48. Na extremidade baixa, o sistema telefônico tem forma de estrela, com todos os circuitos terminais em uma vizinhança convergindo em uma estação final. Ao contrário, a televisão a cabo consiste em um único cabo longo que passa por todas as casas no mesmo bairro. Suponha que um cabo de TV do futuro fosse uma fibra de 10 Gbps, em vez de um fio de cobre. Ele poderia ser usado para simular o modelo de telefonia em que todos têm sua própria linha privada até a estação final? Nesse caso, quantas casas com um telefone poderiam ser conectadas a uma única fibra?

49. Uma empresa de serviços a cabo decide oferecer acesso à Internet por cabo em um bairro que tem 5 mil casas. A empresa utiliza um cabo coaxial e uma alocação de espectro que permite alcançar a largura de banda de 100 Mbps downstream por cabo. Para atrair clientes, a empresa decide garantir pelo menos 2 Mbps de largura de banda downstream para cada casa em qualquer instante. Descreva o que a empresa de serviços a cabo precisa fazer para fornecer essa garantia.

50. Usando a alocação espectral mostrada na Figura 2.46 e as informações dadas no texto, quantos Mbps um sistema de cabo aloca para o tráfego upstream e quantos para o tráfego downstream?

51. Com que velocidade um usuário de cabo recebe dados se a rede estiver ociosa? Suponha que a interface do usuário seja
    (a) Ethernet a 10 Mbps
    (b) Ethernet a 100 Mbps
    (c) Wireless a 54 Mbps.

52. A multiplexação de vários fluxos de dados STS-1, chamados tributários, desempenha um papel importante na SONET. Um multiplexador 3:1 efetua a multiplexação de três tributários STS-1 de entrada em um único fluxo STS-3 de saída. Essa multiplexação é feita byte a byte, isto é, os três primeiros bytes de saída são os primeiros bytes dos tributários 1, 2 e 3, respectivamente. Os três bytes de saída seguintes são os próximos bytes dos tributários 1, 2 e 3, respectivamente e assim por diante. Escreva um programa que simule esse multiplexador 3:1. Seu programa deve consistir em cinco processos. O principal cria quatro processos, um para cada um dos três tributários STS-1 e um processo para o multiplexador. Cada processo tributário lê um quadro STS-1 de um arquivo de entrada como uma sequência de 810 bytes. Eles enviam seus quadros (byte por byte) ao processo multiplexador. Este recebe esses bytes e transmite como saída um quadro STS-3 (byte por byte), gravando-o na saída padrão. Utilize pipes para efetuar a comunicação entre os processos.

53. Escreva um programa para implementar CDMA. Suponha que a extensão de uma sequência de chips seja oito e o número de estações transmitindo seja quatro. Seu programa consiste em três conjuntos de processos: quatro processos transmissores (t0, t1, t2 e t3), um processo de junção e quatro processos receptores (r0, r1, r2 e r3). O programa principal, que também atua como o processo de junção, primeiro lê quatro sequências de chips (notação bipolar) da entrada padrão e uma sequência de 4 bits (1 bit por processo transmissor a ser transmitido), e se bifurca em quatro pares de processos transmissores e receptores. Cada par de processos transmissor/receptor (t0,r0; t1,r1; t2,r2; t3,r3) é atribuído a uma sequência de chips e cada processo transmissor é atribuído a um 1 bit (primeiro bit para t0, segundo bit para t1 e assim por diante). Em seguida, cada processo transmissor calcula o sinal a ser transmitido (uma sequência de 8 bits) e o envia para o processo de junção. Depois de receber sinais de todos os processos transmissores, o processo de junção combina os sinais e envia o sinal combinado aos quatro processos receptores. Então, cada processo receptor calcula o bit que recebeu e o imprime na saída-padrão. Use pipes para a comunicação entre processos.

A camada de enlace de dados

3 Capítulo

Neste capítulo, estudaremos os princípios de projeto da segunda camada, a de enlace de dados. Neste estudo, trataremos de algoritmos que permitem uma comunicação eficiente e confiável de unidades de informação inteiras, chamadas quadros (em vez de bits individuais, como na camada física), entre dois computadores adjacentes. Por adjacentes queremos dizer que as duas máquinas estão fisicamente conectadas por meio de um canal de comunicação que funciona conceitualmente como um fio (por exemplo, um cabo coaxial, uma linha telefônica ou um canal sem fio). A propriedade essencial de um canal que o torna semelhante a um fio é o fato de que os bits são entregues na ordem exata em que são enviados.

Em princípio, você poderá pensar que esse problema é tão trivial que não há nada para estudar — a máquina *A* simplesmente coloca os bits no fio e a máquina *B* os retira de lá. Infelizmente, os canais de comunicação algumas vezes produzem erros. Além disso, eles têm uma taxa de dados finita, e há um atraso de propagação diferente de zero entre o momento em que um bit é enviado e o momento em que ele é recebido. Essas limitações têm implicações importantes para a eficiência da transferência de dados. Os protocolos usados para comunicações devem levar todos esses fatores em consideração. Tais protocolos são o assunto deste capítulo.

Após uma introdução às principais questões de projeto presentes na camada de enlace de dados, começaremos nosso estudo dos protocolos dessa camada verificando a natureza dos erros e como eles podem ser detectados e corrigidos. Em seguida, estudaremos uma série de protocolos de complexidade crescente e mostraremos como cada um deles resolve um número cada vez maior de problemas dessa camada. Por fim, concluiremos o capítulo com alguns exemplos de protocolos de enlace de dados.

3.1 Questões de projeto da camada de enlace de dados

A camada de enlace de dados usa os serviços da camada física para enviar e receber bits pelos canais de comunicação. Ela tem diversas funções, entre as quais:

1. fornecer uma interface de serviço bem definida à camada de rede;
2. lidar com erros de transmissão;
3. regular o fluxo de dados de tal forma que receptores lentos não sejam atropelados por transmissores rápidos.

Para alcançar esses objetivos, a camada de enlace de dados recebe os pacotes da camada de rede e os encapsula em **quadros** para transmissão. Cada quadro contém um cabeçalho (header) de quadro, um campo de carga útil, que conterá o pacote, e um final (trailer) de quadro, como mostra a Figura 3.1. O gerenciamento de quadros constitui o núcleo das atividades da camada de enlace de dados. Nas próximas seções, examinaremos em detalhes todas as questões mencionadas.

Embora este capítulo trate explicitamente da camada de enlace de dados e de seus protocolos, muitos dos princípios que estudaremos aqui, como o controle de erros e o controle de fluxo, são encontrados em protocolos de transporte e também em outros protocolos. Isso ocorre porque a confiabilidade é um objetivo geral, e ela é alcançada quando todas as redes funcionam juntas. De fato, em muitas redes, essas funções são encontradas apenas nas camadas superiores, com a camada de enlace de dados realizando a tarefa mínima, que é 'suficientemente boa'. Porém, independentemente do lugar onde elas são encontradas, os princípios são quase idênticos. Na camada de enlace de da-

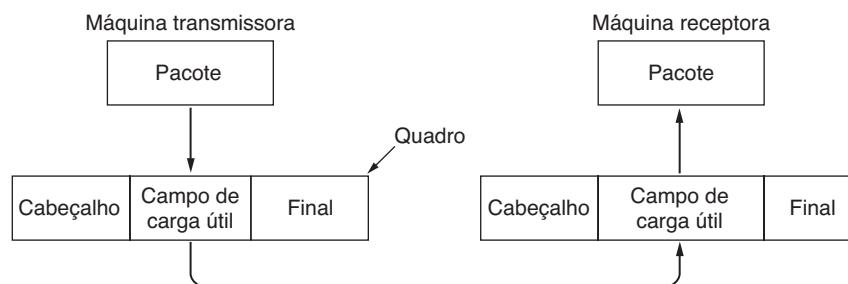



Figura 3.1 | Relacionamento entre pacotes e quadros.

dos, eles surgem com frequência em sua forma mais simples e mais pura, o que faz dessa camada um bom lugar para examiná-los em detalhes.

3.1.1 Serviços oferecidos à camada de rede

A função da camada de enlace de dados é fornecer serviços à camada de rede. O principal serviço é transferir dados da camada de rede da máquina de origem para a camada de rede da máquina de destino. Na camada de rede da máquina de origem há uma entidade, chamada processo, que entrega alguns bits à camada de enlace de dados para transmissão ao destino. A tarefa da camada de enlace de dados é transmitir os bits à máquina de destino, de forma que eles possam ser entregues à camada de rede dessa máquina, como mostra a Figura 3.2(a). A transmissão propriamente dita segue o trajeto descrito na Figura 3.2(b); no entanto, é mais fácil pensar em termos de dois processos da camada de enlace de dados que se comunicam por intermédio de um protocolo de enlace de dados. Por essa razão, utilizaremos implicitamente o modelo da Figura 3.2(a) em todo este capítulo.

A camada de enlace de dados pode ser projetada de modo a oferecer diversos serviços. Os serviços reais oferecidos podem variar de um protocolo para outro. Três possibilidades razoáveis que consideraremos são:

1. serviço não orientado a conexões sem confirmação;
2. serviço não orientado a conexões com confirmação;
3. serviço orientado a conexões com confirmação.

O serviço não orientado a conexões sem confirmação consiste em fazer a máquina de origem enviar quadros independentes à máquina de destino, sem que esta confirme o recebimento desses quadros. A Ethernet é um bom exemplo de uma camada de enlace de dados que oferece essa classe de serviço. Nenhuma conexão lógica é estabelecida antes ou liberada depois do processo. Se um quadro for perdido em decorrência de ruídos na linha, não haverá nenhuma tentativa de detectar a perda ou de recuperá-lo na camada de enlace de dados. Essa classe de serviço é apropriada quando a taxa de erros é muito baixa, e a recuperação fica a cargo das camadas mais altas. Ela também é apropriada para o tráfego em tempo real, no qual, a exemplo da voz, os dados atrasados causam mais problemas que os dados recebidos com falhas.

O próximo passo em termos de confiabilidade é o serviço não orientado a conexões com confirmação. Quando esse serviço é oferecido, ainda não há conexões lógicas sendo usadas, mas cada quadro enviado é confirmado individualmente. Dessa forma, o transmissor sabe se um quadro chegou corretamente ou não. Caso não tenha chegado dentro de um intervalo específico, o quadro poderá ser enviado outra vez. Esse serviço é útil em canais não confiáveis, como os sistemas sem fio. O padrão 802.11 (WiFi) é um bom exemplo dessa classe de serviço.

Talvez valha a pena destacar que oferecer recursos de confirmação no nível da camada de enlace de dados é uma questão de otimização, nunca uma exigência. A camada de rede sempre pode enviar um pacote e esperar que ele seja confirmado por seu par na máquina remota. Se a confirmação não chegar durante o intervalo do timer, o transmissor poderá enviar a mensagem inteira mais uma vez. O problema dessa estratégia é que ela pode ser ineficaz. Os enlaces normalmente têm um quadro com comprimento máximo estrito imposto pelo hardware e atrasos de propagação conhecidos. A camada de rede não conhece esses parâmetros. Ela pode enviar um pa-

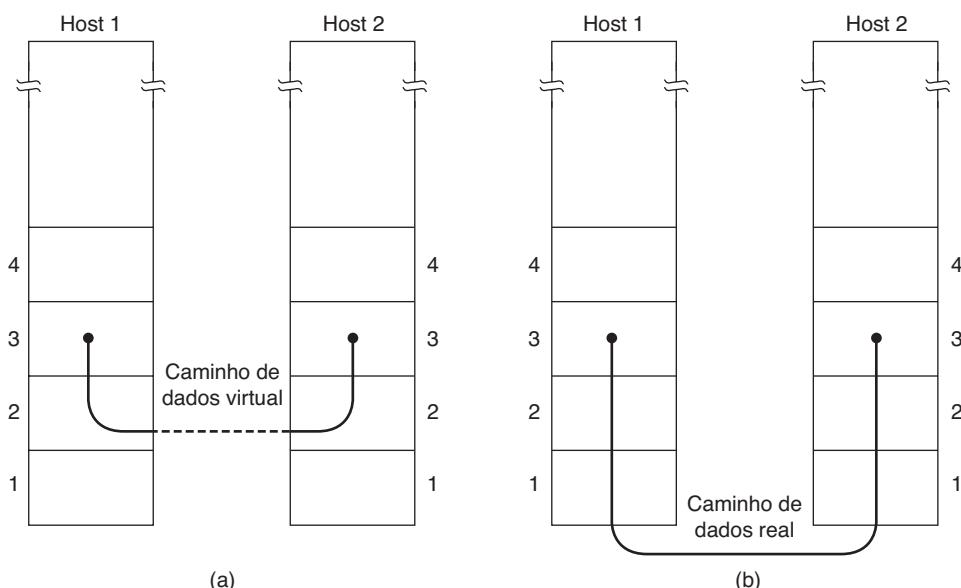



Figura 3.2 | (a) Comunicação virtual. (b) Comunicação real.

cote grande subdividido em, digamos, dez quadros, dos quais dois são perdidos, em média. O tempo necessário para efetivar a transmissão do pacote com sucesso poderá ser muito longo. Em vez disso, se quadros individuais forem confirmados e retransmitidos, então os erros podem ser corrigidos mais diretamente e mais rapidamente. Em canais confiáveis, como a fibra óptica, o overhead de um protocolo de enlace de dados muito sofisticado talvez seja desnecessário, mas, em canais sem fio (com sua inerente falta de confiabilidade), o custo compensa.

Voltando aos nossos serviços, o serviço mais sofisticado que a camada de enlace de dados é capaz de oferecer à camada de rede é o serviço orientado a conexões. Com ele, as máquinas de origem e destino estabelecem uma conexão antes de qualquer dado ser transferido. Cada quadro enviado pela conexão é numerado, e a camada de enlace de dados garante que cada quadro será, de fato, recebido. Além disso, essa camada garante que todos os quadros serão recebidos uma única vez e na ordem correta. Assim, os serviços orientados a conexões fornecem aos processos da camada de rede o equivalente a um fluxo de bits confiável. Isso é apropriado para enlaces longos, não confiáveis, como um canal de satélite ou um circuito telefônico interurbano. Se o serviço não orientado a conexões com confirmação fosse usado, é possível imaginar que as confirmações perdidas poderiam fazer com que um quadro fosse enviado e recebido várias vezes, desperdiçando largura de banda.

Quando o serviço orientado a conexões é usado, as transferências passam por três fases distintas. Na primeira fase, a conexão é estabelecida, fazendo ambos os lados inicializar as variáveis e os contadores necessários para controlar os quadros que são recebidos e os que não são. Na segunda fase, um ou mais quadros são realmente transmitidos. Na terceira e última fase, a conexão é desfeita, liberando-se as variáveis, os buffers e os outros recursos usados para mantê-la.

3.1.2 Enquadramento

Para oferecer serviços à camada de rede, a camada de enlace de dados deve usar o serviço fornecido a ela pela camada física. O que a camada física faz é aceitar um fluxo de bits brutos e tentar entregá-lo ao destino. Se o canal tiver ruído, como acontece na maioria dos enlaces sem fio e em alguns enlaces com fio, a camada física acrescentará alguma redundância aos seus sinais, para reduzir a taxa de erros de bits para um nível tolerável. Contudo, o fluxo de bits recebido pela camada de enlace de dados não tem garantia de estar livre de erros. Alguns bits podem ter valores diferentes e o número de bits recebidos pode ser menor, igual ou maior que o número de bits transmitidos. A camada de enlace de dados é responsável por detectar e, se necessário, corrigir erros.

Em geral, a estratégia adotada pela camada de enlace de dados é dividir o fluxo de bits em quadros distintos, calcular um pequeno valor (um token), chamado de checksum (somatório de verificação), para cada quadro e incluir essa soma de verificação no quadro quando ele for transmitido. (Os algoritmos de checksum serão discutidos mais adiante neste capítulo.) Quando um quadro chega a seu destino, o checksum é recalculado. Se o checksum recém-calculado for diferente do que está contido no quadro, a camada de enlace de dados saberá que houve um erro e tomará providências para lidar com ele (por exemplo, descartando o quadro defeituoso e possivelmente também enviando de volta um relatório de erros).

A divisão do fluxo de bits em quadros é mais difícil do que parece à primeira vista. Um bom projeto deve tornar fácil para um receptor encontrar o início de novos quadros enquanto usa pouca largura de banda. Nesta seção, examinaremos quatro métodos:

1. contagem de caracteres;
2. bytes de flag com inserção de bytes;
3. flags iniciais e finais, com inserção de bits;
4. violações de codificação da camada física.

O primeiro método de enquadramento utiliza um campo no cabeçalho para especificar o número de bytes no quadro. Quando vê a contagem de caracteres, a camada de enlace de dados de destino sabe quantos bytes devem vir em seguida e, consequentemente, onde está o fim do quadro. Essa técnica é mostrada na Figura 3.3(a) para quatro pequenos quadros, como exemplo, de tamanhos 5, 5, 8 e 8 bytes, respectivamente.

O problema com esse algoritmo é que a contagem pode ser adulterada por um erro de transmissão. Por exemplo, se a contagem 5 no segundo quadro da Figura 3.3(b) se tornar 7, em virtude da inversão de um único bit, o destino perderá a sincronização e não será capaz de localizar o início do quadro seguinte. Mesmo que o checksum esteja incorreto, de modo que o destino saiba que o quadro está defeituoso, ele ainda não terá informações suficientes para saber onde começa o quadro seguinte. Enviar um quadro de volta à origem solicitando retransmissão também não ajuda, pois o destino não sabe quantos caracteres deverão ser ignorados para chegar ao início da retransmissão. Por essa razão, o método de contagem de caracteres quase não é mais usado.

O segundo método de enquadramento contorna o problema de ressincronização após um erro, fazendo cada quadro começar e terminar com bytes especiais. Normalmente o mesmo byte, chamado **byte de flag**, é usado como delimitador de início e de fim, como mostra a Figura 3.4(a), na qual ele é representado por FLAG. Dois bytes de flag consecutivos indicam o fim de um quadro e o início do próximo. Assim, se o receptor perder a sincronização, ele poderá simplesmente procurar dois bytes de flag para encontrar o final do quadro atual e o início do seguinte.

Entretanto, ainda existe um problema sério que precisamos resolver. É bem possível que o byte de flag ocorra

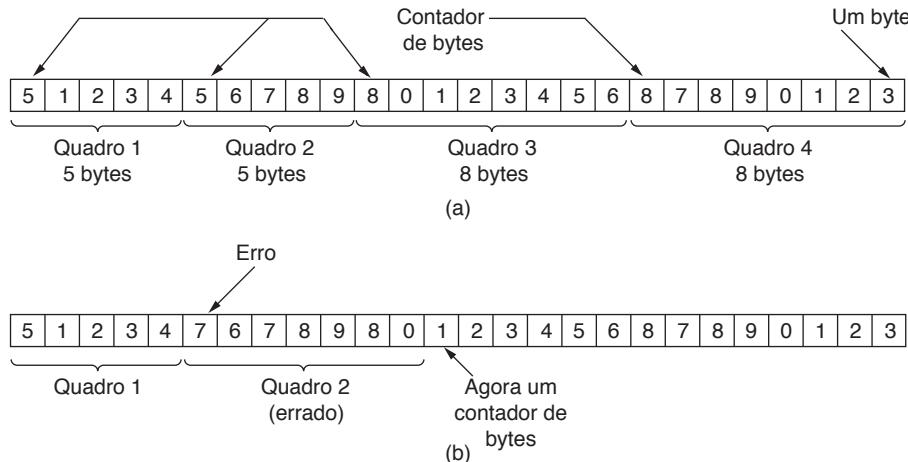



Figura 3.3 | Um fluxo de caracteres. (a) Sem erros. (b) Com um erro.

nos dados, especialmente quando são transmitidos dados binários, como fotografias ou músicas. Essa situação poderia interferir no enquadramento. Uma forma de solucionar esse problema é fazer com que a camada de enlace de dados do transmissor inclua um caractere de escape especial (ESC) imediatamente antes de cada byte de flag "acidental" nos dados. Assim, o byte de flag de enquadramento pode ser distinguido daquele nos dados pela ausência ou presença de um byte de escape antes dele. A camada de enlace de dados na extremidade receptora remove o byte de escape antes de entregar os dados à camada de rede. Essa técnica é chamada **inserção de bytes** (**byte stuffing**).

É claro que a próxima pergunta é: o que acontecerá se um byte de escape ocorrer em uma posição intermediária dos dados? Nesse caso, ele também será preenchido com um byte de escape. No receptor, o primeiro byte de escape é removido, deixando o byte de dados seguinte (que poderia ser outro byte de escape ou o byte de flag). Alguns exemplos são mostrados na Figura 3.4(b). Em todos os casos, a sequência de bytes entregue após a remoção dos bytes inseridos é exatamente igual à sequência de bytes original. Ainda podemos procurar por um limite de quadro buscando dois bytes de flag em sequência, sem nos preocuparmos em desfazer os escapes.

O esquema de inserção de bytes representado na Figura 3.4 é uma ligeira simplificação do que é utilizado no protocolo ponto a ponto, ou **PPP** (**Point-to-Point Protocol**), usado para carregar pacotes por enlaces de comunicação. Descreveremos o PPP mais no final deste capítulo.

O terceiro método de delimitação do fluxo de bits contorna uma desvantagem da inserção de bytes, ou seja, o

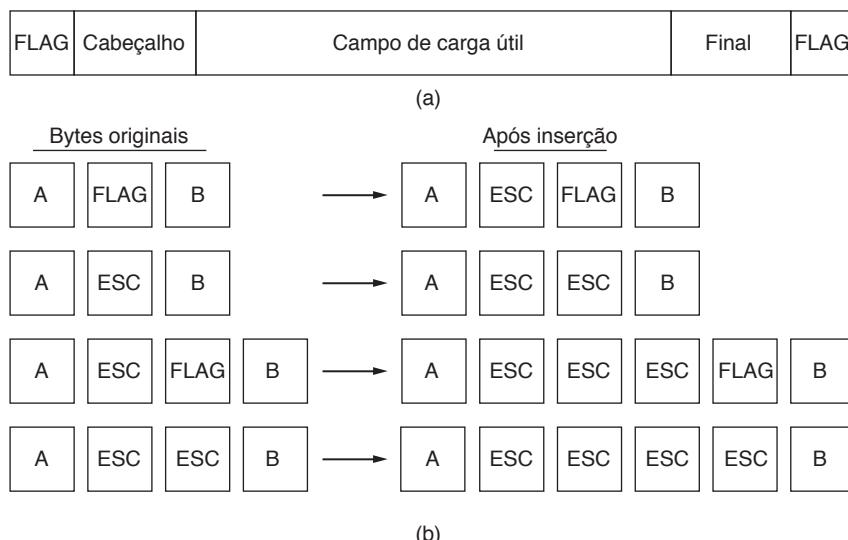



Figura 3.4 | (a) Quadro delimitado por bytes de flag. (b) Quatro exemplos de sequências de bytes, antes e depois da inserção de bytes.

fato de ela estar ligada ao uso de bytes de 8 bits. O enquadramento também pode ser feito em nível de bit, de modo que os quadros podem conter um número qualquer de bits, compostos de unidades de qualquer tamanho. Ele foi desenvolvido para o então muito popular protocolo de controle de enlace de dados de alto nível, ou **HDLC** (**High-level Data Link Control**). Cada quadro começa e termina com um padrão de bits especial, 01111110, ou 0x7E em hexadecimal (na verdade, um byte de flag). Sempre que encontra cinco valores 1 consecutivos nos dados, a camada de enlace de dados do transmissor automaticamente insere um bit 0 no fluxo de bits que está sendo enviado. Essa **inserção de bits** é semelhante à inserção de bytes, na qual um byte de escape é inserido no fluxo de caracteres enviado antes de ocorrer um byte de flag nos dados. Isso também garante uma densidade mínima de transições, o que ajuda a camada física a manter a sincronização. O USB (Universal Serial Bus) utiliza a inserção de bits por esse motivo.

Ao ver cinco bits 1 consecutivos sendo recebidos, seguidos por um bit 0, o receptor automaticamente remove o bit 0. A inserção de bits, assim como a inserção de bytes, é completamente transparente para a camada de rede de ambos os computadores. Se os dados do usuário contiverem o padrão de flag 01111110, esse flag será transmitido como 011111010, mas será armazenado na memória do receptor como 011111110. A Figura 3.5 mostra um exemplo de inserção de bits.

Com a inserção de bits, o limite entre dois quadros pode ser reconhecido sem nenhum tipo de ambiguidade pelo padrão de flags. Desse modo, se o receptor perder o controle de onde estão os dados, bastará varrer a entrada em busca de sequências de flags, uma vez que elas nunca ocorrem dentro dos dados, apenas nos limites dos quadros.

Com a inserção de bits e de bytes, um efeito colateral é que o comprimento de um quadro agora depende do conteúdo dos dados que ele carrega. Por exemplo, se não houver bytes de flag nos dados, 100 bytes podem ser transportados em um quadro de aproximadamente 100 bytes. Porém, se os dados consistirem unicamente de bytes de flag, cada um terá um bit de escape associado e o quadro terá aproximadamente 200 bytes de comprimento. Com a inserção de bits, o aumento seria de aproximadamente 12,5%, pois 1 bit é inserido a cada byte.

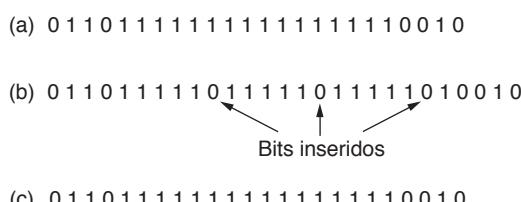



Figura 3.5 | Inserção de bits. (a) Os dados originais. (b) Como os dados são exibidos na linha. (c) Como os dados são armazenados na memória do receptor após a remoção de bits.

O último método de enquadramento é usar um atalho da camada física. No Capítulo 2, vimos que a codificação de bits como sinais normalmente inclui redundância para ajudar o receptor. Essa redundância significa que alguns sinais não ocorrerão em dados regulares. Por exemplo, no código de linha 4B/5B, 4 bits de dados são mapeados para 5 bits de sinal, para garantir transições de bits suficientes. Isso significa que 16 das 32 possibilidades de sinal não são usadas. Podemos usar alguns sinais reservados para indicar o início e o final dos quadros. Com efeito, estamos usando 'violações de código' para delimitar os quadros. A beleza desse esquema é que, por serem sinais reservados, é fácil encontrar o início e o final dos quadros, e não é preciso inserir bits nos dados.

Muitos protocolos de enlace de dados, por segurança, usam uma combinação desses métodos. Um padrão comum usado para Ethernet e 802.11 é fazer com que um quadro comece com um padrão bem definido, chamado **preâmbulo**. Esse padrão pode ser muito longo (72 bits é típico para redes 802.11), para permitir que o receptor se prepare para um pacote que está chegando. O preâmbulo é, então, seguido por um campo de comprimento (ou seja, um contador) no cabeçalho, que é usado para localizar o final do quadro.

3.1.3 Controle de erros

Após resolvermos o problema da delimitação do início e do fim de cada quadro, vamos ao problema seguinte: como ter certeza de que todos os quadros serão entregues na camada de rede de destino e na ordem apropriada? Suponha, para o momento, que o receptor possa saber se um quadro que ele recebe contém informações corretas ou defeituosas (veremos os códigos usados para detectar e corrigir erros de transmissão na Seção 3.2). Para serviços não orientados a conexões, sem confirmação, pode ser suficiente que o emissor apenas continue enviando quadros sem se importar se eles chegaram corretamente, mas sem dúvida essa não seria uma boa opção para serviços orientados a conexões confiáveis.

A forma mais comum de garantir uma entrega confiável é dar ao transmissor algum tipo de feedback sobre o que está acontecendo no outro extremo da linha. Normalmente, o protocolo solicita que o receptor retorne quadros de controle especiais com confirmações positivas ou negativas sobre os quadros recebidos. Se receber uma confirmação positiva sobre um quadro, o transmissor saberá que o quadro chegou em segurança ao destino. Por outro lado, uma confirmação negativa significa que algo saiu errado e o quadro deve ser retransmitido.

Uma complicação adicional decorre da possibilidade de problemas de hardware fazerem com que um quadro desapareça completamente (por exemplo, em uma rajada de ruídos). Nesse caso, o receptor não reagirá de forma alguma, pois não há motivo para isso. De modo semelhante, se

o quadro de confirmação se perder, o emissor não saberá como prosseguir. Deve ficar claro que um protocolo no qual o transmissor envia um quadro e depois espera por uma confirmação, positiva ou negativa, permanecerá suspenso para sempre caso um quadro tenha sido completamente perdido — por exemplo, em consequência de mau funcionamento do hardware ou canal de comunicação deficiente.

Essa possibilidade é tratada com a introdução de timers na camada de enlace de dados. Quando o transmissor envia um quadro, em geral ele também inicializa um timer. Este é ajustado para ser desativado após um intervalo suficientemente longo para o quadro chegar ao destino, ser processado ali e ter sua confirmação enviada de volta ao transmissor. Em geral, o quadro será recebido de forma correta e a confirmação voltará antes que se alcance o tempo-limite do timer e, nesse caso, ele será cancelado.

No entanto, se a confirmação ou o quadro se perder, o timer será desativado, alertando o transmissor para um problema potencial. A solução óbvia é simplesmente transmitir o quadro outra vez. Entretanto, quando os quadros podem ser transmitidos várias vezes, existe o perigo de o receptor aceitar o mesmo quadro duas ou mais vezes e de repassá-lo à camada de rede mais de uma vez. Para evitar que isso aconteça, geralmente é preciso atribuir números de sequência aos quadros transmitidos, de modo que o receptor possa distinguir as retransmissões dos originais.

A questão do gerenciamento dos timers e dos números de sequência para garantir que cada quadro seja realmente passado para a camada de rede no destino exatamente uma vez, nem mais nem menos, é uma parte importante das atribuições da camada de enlace de dados (e das camadas mais altas). Mais adiante neste capítulo, veremos uma série de exemplos cada vez mais sofisticados, para entender como é feito esse gerenciamento.

3.1.4 Controle de fluxo

Outro aspecto de projeto importante que ocorre na camada de enlace de dados (e também nas camadas mais altas) é o que fazer com um transmissor que sistematicamente deseja transmitir quadros mais rápido do que o receptor pode aceitá-los. Essa situação pode ocorrer quando o transmissor está rodando em um computador rápido e poderoso e o receptor está rodando em uma máquina lenta e inferior. Uma situação comum é quando um smartphone solicita uma página Web de um servidor muito mais poderoso, que abre a mangueira de incêndio e jorra os dados para o pobre e infeliz telefone, até que esteja completamente inundado. Mesmo que a transmissão não tenha erros, o receptor pode não conseguir lidar com os quadros com a rapidez com que chegam, e perderá alguns deles.

Sem dúvida, algo deve ser feito para impedir que essa situação ocorra. São usadas comumente duas abordagens. Na primeira, chamada **controle de fluxo baseado em feedback**, o receptor envia de volta ao transmissor informações que permitam a ele enviar mais dados, ou que pelo menos mostrem ao transmissor a situação real do receptor. Na segunda, chamada **controle de fluxo baseado na velocidade**, o protocolo tem um mecanismo interno que limita a velocidade com que os transmissores podem enviar os dados, sem usar o feedback do receptor.

Neste capítulo, estudaremos os esquemas de controle de fluxo baseado em feedback, principalmente porque os esquemas baseados na velocidade são vistos apenas como parte da camada de transporte (Capítulo 5). Os esquemas baseados em feedback são vistos na camada de enlace e em camadas superiores. O último caso é mais comum hoje em dia, em que o hardware da camada de enlace é projetado para trabalhar com rapidez suficiente para não causar perda. Por exemplo, considera-se às vezes que as implementações de hardware da camada de enlace, como as **placas de interface de rede**, ou **NICs** (**Network Interface Cards**), atuam na "velocidade do fio", o que significa que podem tratar de quadros tão rapidamente quanto eles chegam ao enlace. Portanto, qualquer overrun não será um problema de enlace, já que é tratado por camadas mais altas.

Existem diversos esquemas de controle de fluxo. No entanto, a maioria deles utiliza o mesmo princípio básico. O protocolo contém regras bem definidas sobre quando um transmissor pode enviar o quadro seguinte. Com frequência, essas regras impedem que os quadros sejam enviados até que o receptor tenha concedido permissão para a transmissão, implícita ou explicitamente. Por exemplo, quando uma conexão é estabelecida, o receptor pode informar: "Você pode me enviar *n* quadros agora, mas, depois que eles tiverem sido enviados, não envie mais nada até ser informado de que deve prosseguir". Examinaremos os detalhes em breve.

3.2 Detecção e correção de erros

Vimos no Capítulo 2 que os canais de comunicação possuem diversas características. Alguns canais, como a fibra óptica nas redes de telecomunicações, possuem taxas de erro muito pequenas, de modo que os erros de transmissão são uma ocorrência rara. Mas outros canais, especialmente enlaces sem fios e antigos circuitos terminais, possuem taxas de erro muito maiores. Para esses enlaces, os erros de transmissão são a norma. Eles não podem ser evitados a um custo razoável em termos de desempenho. A conclusão é que os erros de transmissão estão aqui para ficar. O que precisamos aprender é como lidar com eles.

Os projetistas de redes desenvolveram duas estratégias básicas para lidar com os erros. Ambas acrescentam informações redundantes aos dados enviados. Uma estratégia é incluir informações redundantes suficientes para permitir que o receptor deduza quais foram os dados transmitidos. A outra é incluir apenas a redundância suficiente para permitir que o receptor deduza que houve um erro (mas não

qual erro) e solicite uma retransmissão. A primeira estratégia usa **códigos de correção de erros**, enquanto a segunda usa **códigos de detecção de erros**. O uso de códigos de correção de erros normalmente é conhecido como **correção adiantada de erros**, ou **FEC** (**Forward Error Correction**).

Cada uma dessas técnicas ocupa um nicho em ambientes específicos. Em canais altamente confiáveis, como os de fibra, é mais econômico utilizar um código de detecção de erros e simplesmente retransmitir o bloco defeituoso ocasional. Porém, em canais como enlaces sem fios, que geram muitos erros, é melhor adicionar redundância suficiente a cada bloco para que o receptor seja capaz de descobrir qual era o bloco transmitido originalmente. A FEC é usada em canais com ruído porque as retransmissões podem ter erros tanto quanto a primeira transmissão.

Uma consideração importante para esses códigos é o tipo de erro que provavelmente ocorrerá. Nem os códigos de correção de erro nem os de detecção de erro podem lidar com todos os erros possíveis, pois os bits redundantes que oferecem proteção provavelmente serão recebidos com erro, como os bits de dados (o que pode comprometer sua proteção). Seria bom se o canal tratasse os bits redundantes de forma diferente da que trata os bits de dados, mas isso não acontece. Eles todos são apenas bits para o canal. Isso significa que, para evitar erros não detectados, o código precisa ser forte o suficiente para lidar com os erros esperados.

Um modelo é aquele em que os erros são causados por valores extremos de ruído térmico, que abafa o sinal rápida e ocasionalmente, fazendo surgir erros de único bit isolados. Outro modelo consiste em erros que tendem a vir em rajadas, em vez de isolados. Esse modelo vem dos processos físicos que os geram — como uma atenuação profunda em um canal sem fios ou interferência elétrica transitória em um canal com fios.

Os dois modelos importam na prática e possuem diferentes dilemas. O fato de os erros acontecerem em rajadas tem vantagens e desvantagens em relação aos erros isolados de único bit. Uma vantagem é que os dados de computadores sempre são enviados em blocos de bits. Suponha que o tamanho do bloco seja 1.000 bits e que a taxa de erros seja 0,001 por bit. Se os erros fossem independentes, a maioria dos blocos teria um erro. Porém, se os erros surgirem em rajadas de 100, apenas um ou dois blocos em 100 será(ão) afetado(s), em média. A desvantagem dos erros em rajada é que, quando ocorrem, são muito mais difíceis de corrigir que os erros isolados.

Também existem outros tipos de erros. Às vezes, o local de um erro será conhecido, talvez porque a camada física tenha recebido um sinal analógico que foi longe do valor esperado para 0 ou 1 e declarado que o bit foi perdido. Essa situação é chamada **canal de apagamento**. É mais fácil corrigir erros em canais de apagamento do que em canais que invertem bits, pois, mesmo que o valor do bit tenha sido perdido, pelo menos sabemos qual deles tem erro. Contudo, normalmente não temos o benefício dos apagamentos.

Em seguida, examinaremos os códigos de correção de erros e os códigos de detecção de erros. Porém, por favor, tenha dois pontos em mente. Primeiro, cobrimos esses códigos na camada de enlace porque esse é o primeiro lugar em que deparamos com o problema de transmitir grupos de bits de modo confiável. Porém, os códigos são muito usados porque a confiabilidade é uma preocupação geral. Os códigos de correção de erros também são vistos na camada física, principalmente para canais com ruído e em camadas superiores, particularmente para mídia em tempo real e distribuição de conteúdo. Os códigos de detecção de erros normalmente são usados nas camadas de enlace, rede e transporte.

O segundo ponto que se deve ter em mente é que os códigos de detecção e/ou correção de erros são matemática aplicada. A menos que você seja particularmente adepto aos campos de Galois ou às propriedades das matrizes esparsas, deverá obter códigos com boas propriedades a partir de uma fonte confiável, em vez de criar os seus próprios. De fato, é isso que muitos padrões de protocolo fazem, com os mesmos códigos aparecendo por repetidas vezes. No material a seguir, estudaremos um código simples com detalhes e depois descreveremos rapidamente os códigos avançados. Desse modo, podemos entender os dilemas a partir do código simples e falar sobre os códigos usados na prática por meio dos códigos avançados.

3.2.1 Códigos de correção de erros

Vamos examinar quatro tipos de código de correção de erros:

1. códigos de Hamming;
2. códigos de convolução binários;
3. códigos de Reed-Solomon;
4. códigos de verificação de paridade de baixa densidade.

Todos esses códigos acrescentam redundância às informações enviadas. Um quadro consiste em m bits de dados (ou seja, mensagem) e r bits redundantes (ou seja, verificação). Em um **código de bloco**, os r bits de verificação são calculados unicamente como uma função dos m bits de dados com os quais eles são associados, como se os m bits fossem pesquisados em uma grande tabela para encontrar seus r bits de verificação correspondentes. Em um **código sistemático**, os m bits de dados são enviados diretamente, com os bits de verificação, em vez de ser codificados eles mesmos antes de ser enviados. Em um **código linear**, os r bits de verificação são calculados como uma função linear dos m bits de dados. A operação OU exclusivo (XOR), ou adição de módulo 2, é uma escolha popular. Isso significa que a codificação pode ser feita com operações como

multiplicações de matrizes ou circuitos lógicos simples. Os códigos que veremos nesta seção são códigos de bloco lineares, sistemáticos, a menos que sejam observados de outra forma.

Considere que o tamanho total de um bloco seja n (ou seja, $n = m + r$). Descreveremos isso como um código (n,m). Uma unidade de n bits contendo bits de dados e verificação é conhecida como uma **palavra de código** de n bits. A **taxa de código**, ou simplesmente taxa, é a fração da palavra de código que transporta informações não redundantes, ou m/n. As taxas usadas na prática variam bastante. Elas poderiam ser 1/2 para um canal com ruído, em que metade da informação recebida é redundante, ou perto de 1 para um canal de alta qualidade, com apenas um pequeno número de bits de verificação acrescentados a uma mensagem grande.

Para entender como os erros podem ser tratados, primeiro é necessário examinar de perto o que realmente é um erro. Dadas duas palavras de código que podem ser transmitidas ou recebidas — digamos, 10001001 e 10110001 —, é possível determinar quantos bits correspondentes diferem. Nesse caso, são 3 os bits divergentes. Para determinar quantos bits apresentam diferenças, basta efetuar uma operação XOR entre as duas palavras de código e contar o número de bits 1 no resultado. Por exemplo:

$$\begin{array}{l} 10001001 \\ \underline{10110001} \\ 00111000 \end{array}$$

O número de posições de bits em que duas palavras de código diferem entre si é chamado **distância de Hamming** (Hamming, 1950). Isso significa que, se duas palavras de código estiverem a uma distância de Hamming uma da outra igual a d, será necessário corrigir d erros de bits isolados para converter uma palavra na outra.

Dado o algoritmo para calcular os bits de verificação, é possível construir uma lista completa das palavras de código válidas e, a partir dessa lista, encontrar as duas palavras de código com a menor distância de Hamming. Essa é a distância de Hamming do código completo.

Na maioria das aplicações de transmissão de dados, todas as 2^m mensagens de dados possíveis são válidas; no entanto, em virtude da forma como os bits de verificação são calculados, nem todas as 2^n palavras de código possíveis são usadas. De fato, quando existem r bits de verificação, somente a pequena fração de $2^m/2^n$ ou $1/2^r$ das mensagens possíveis será uma palavra de código válida. É a dispersão com que a mensagem é embutida no espaço das palavras de código que permite que o receptor detecte e corrija erros.

As propriedades de detecção e de correção de erros de um código dependem de sua distância de Hamming. Para detectar d erros de modo confiável, você precisa de um código de distância $d + 1$, pois com tal código não há como d erros de bits transformarem uma palavra de código válida em outra. Ao detectar uma palavra de código inválida, isso pode indicar que houve um erro de transmissão. Da mesma forma, para corrigir d erros, você precisa de um código de distância $2d + 1$ porque, dessa forma, as palavras de código válidas estarão tão distantes que, mesmo com d alterações, a palavra de código original continuará mais próxima do que qualquer outra. Isso significa que a palavra de código original pode ser determinada exclusivamente com base na suposição de que um número maior de erros é menos provável.

Como um exemplo simples de código de correção de erros, considere um código que contenha apenas quatro palavras de código válidas:

0000000000, 0000011111, 1111100000 e 1111111111

Esse código tem uma distância igual a 5, o que significa que ele pode corrigir erros duplos ou detectar erros quádruplos. Se a palavra de código 0000000111 for detectada e esperarmos apenas erros de 1 ou 2 bits, o receptor saberá que a original deve ter sido 0000011111. No entanto, se um erro triplo transformar 0000000000 em 0000000111, o erro não será corrigido da maneira adequada. Alternativamente, se esperarmos todos esses erros, podemos detectá-los. Como nenhuma das palavras de código recebidas é uma palavra de código válida, um erro deve ter ocorrido. Deve ficar evidente que, nesse exemplo, não podemos corrigir erros duplos e detectar erros quádruplos, pois isso exigiria que interpretássemos uma palavra de código recebida de duas maneiras.

Em nosso exemplo, a tarefa de decodificar encontrando a palavra de código válida mais próxima da palavra de código recebida pode ser feita por inspeção. Infelizmente, no caso mais geral, quando todas as palavras de código precisam ser avaliadas como candidatas, essa tarefa pode ser uma busca demorada. Em vez disso, os códigos práticos são criados de modo que admitam atalhos para encontrar o que provavelmente foi a palavra de código original.

Suponha que desejemos criar um código com m bits de mensagem e r bits de verificação que permitirão a correção de todos os erros simples. Cada uma das 2^m mensagens válidas tem n palavras de código inválidas a uma distância da mensagem igual a 1. Essas palavras inválidas são formadas pela inversão sistemática de cada um dos n bits da palavra de código de n bits formada a partir dela. Portanto, cada uma das 2^m mensagens válidas exige $n + 1$ padrões de bits dedicados a ela. Como o número total de padrões de bits é 2^n, devemos ter $(n + 1)2^m \leq 2^n$. Utilizando $n = m + r$, esse requisito passa a ser

$$(m + r + 1) \leq 2^r \qquad (3.1)$$

Se m for determinado, o limite para o número de bits de verificação necessários para corrigir erros isolados será mais baixo.

Esse limite teórico mais baixo pode, na verdade, ser alcançado pela utilização de um método criado por Hamming (1950). Nos **códigos de Hamming**, os bits da palavra de código são numerados consecutivamente, começando com o bit 1 da extremidade esquerda, o bit 2 imediatamente à sua direita e assim por diante. Os bits que são potências de 2 (1, 2, 4, 8, 16 etc.) são bits de verificação. Os outros (3, 5, 6, 7, 9 etc.) são preenchidos com os *m* bits de dados. Esse padrão pode ser visto para um código de Hamming (11,7) com 7 bits de dados e 4 bits de verificação na Figura 3.6. Cada bit de verificação força a paridade de algum conjunto de bits, incluindo seu próprio conjunto, a ser par (ou ímpar). Um bit pode ser incluído em vários cálculos de verificação de bits. Se quiser ver para quais bits de verificação o bit de dados na posição *k* contribui, reescreva *k* como uma soma de potências de 2. Por exemplo, $11 = 1 + 2 + 8$ e $29 = 1 + 4 + 8 + 16$. Um bit é verificado apenas por aqueles bits de verificação que ocorrem em sua expansão (por exemplo, o bit 11 é verificado pelos bits 1, 2 e 8). No exemplo, os bits de verificação são calculados para somas de paridade par para uma mensagem que é a letra ASCII 'A'.

Essa construção resulta em um código com uma distância de Hamming igual a 3, o que significa que ela pode corrigir erros simples (ou detectar erros duplos). O motivo para a numeração muito cuidadosa dos bits de mensagem e verificação torna-se aparente no processo de decodificação. Quando uma palavra de código chega, o receptor refaz os cálculos do bit de verificação, incluindo os valores dos bits de verificação recebidos. Chamamos estes de resultados de verificação. Se os bits de verificação forem corretos, então, para somas de paridade par, cada resultado de verificação deve ser zero. Nesse caso, a palavra de código é aceita como válida.

Entretanto, se os resultados da verificação não forem todos zero, um erro foi detectado. O conjunto de resultados de verificação forma a **síndrome de erro** que é usada para localizar e corrigir o erro. Na Figura 3.6, um erro de único bit ocorreu no canal, de modo que os resultados de verificação são 0, 1, 0 e 1 para $k = 8$, 4, 2 e 1, respectivamente. Isso gera uma síndrome de 0101 ou $4 + 1 = 5$. Pelo projeto do esquema, isso significa que o quinto bit está com erro. A inversão do bit incorreto (que pode ser um bit de verificação ou de dados) e o descarte dos bits de verificação geram a mensagem correta de um ASCII 'A'.

As distâncias de Hamming são valiosas para entender os códigos de bloco, e os códigos de Hamming são usados na memória de correção de erro. Contudo, a maioria das redes usa códigos mais fortes. O segundo código que veremos é um **código de convolução**. Este é o único que veremos que não é um código de bloco. Em um código de convolução, um codificador processa uma sequência de bits de entrada e gera uma sequência de bits de saída. Não existe tamanho de mensagem ou limite de codificação natural, como em um código de bloco. A saída depende dos bits de entrada atual e anterior. Ou seja, o codificador tem memória. O número de bits anteriores do qual a saída depende é chamado **comprimento de restrição** do código. Os códigos de convolução são especificados em termos de sua taxa e comprimento de restrição.

Os códigos de convolução são muito usados em redes implantadas, por exemplo, como parte do sistema de telefonia móvel GSM, em comunicações por satélite e nas redes 802.11. Como exemplo, um código de convolução popular aparece na Figura 3.7. Esse código é conhecido como código de convolução da NASA, com $r = 1/2$ e $k = 7$, pois ele foi usado inicialmente para as missões espaciais Voyager, a partir de 1977. Desde então, ele tem sido bastante reutilizado, por exemplo, como parte do padrão 802.11.

Na Figura 3.7, cada bit de entrada no lado esquerdo produz dois bits de saída no lado direito, os quais são somas de operações XOR entre a entrada e o estado interno. Por lidar com bits e realizar operações lineares, esse é um código de convolução binário, linear. Como 1 bit de entrada produz 2 bits de saída, o código é 1/2. Ele não é sistemático, pois nenhum dos bits de saída é simplesmente o bit de entrada.

O estado interno é mantido em seis registradores de memória. Toda vez que outro bit é inserido, os valores nos registradores são deslocados para a direita. Por exemplo, se 111 for inserido e o estado inicial contiver zeros, o estado interno, escrito da esquerda para a direita, se tornará 100000, 110000 e 111000 após o primeiro, o segundo e o terceiro bits ser inseridos. Os bits de saída serão 11, seguidos por 10 e, depois, 01. São necessários sete deslocamentos para esvaziar uma entrada completamente, de modo

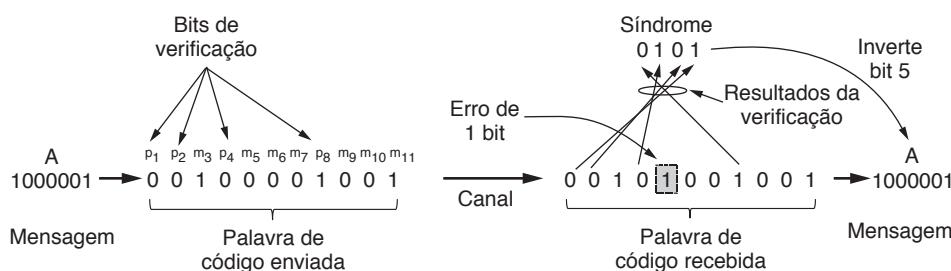



Figura 3.6 | Exemplo de um código de Hamming (11,7) corrigindo um erro de único bit.

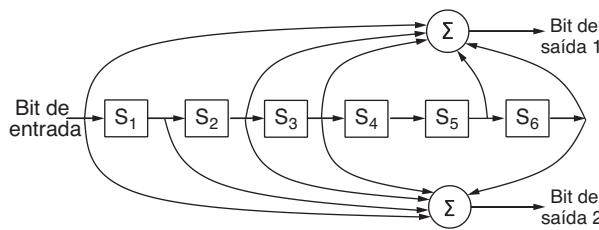



Figura 3.7 | O código de convolução binário da NASA usado no padrão 802.11.

que ela não afete a saída. O comprimento da restrição desse código é, portanto, $k = 7$.

Um código de convolução é decodificado encontrando-se a sequência de bits de entrada que tem maior probabilidade de ter produzido a sequência observada de bits de saída (que inclui quaisquer erros). Para valores pequenos de k, isso é feito com um algoritmo bastante usado, desenvolvido por Viterbi (Forney, 1973). O algoritmo percorre a sequência observada, mantendo para cada etapa e para cada estado interno possível a sequência de entrada que teria produzido a sequência observada com o mínimo de erros. A sequência de entrada que exige o mínimo de erros no final é a mensagem mais provável.

Os códigos de convolução têm sido populares na prática porque é fácil fatorar a incerteza de um bit sendo 0 ou 1 na decodificação. Por exemplo, supondo que –1V seja o nível lógico 0 e +1V seja o nível lógico 1, poderíamos receber 0,9V e –0,1V para 2 bits. Em vez de mapear esses sinais como 1 e 0 imediatamente, gostaríamos de tratar 0,9V como "provavelmente 1" e –0,1V como "talvez 0" e corrigir a sequência como um todo. As extensões do algoritmo de Viterbi podem trabalhar com essas incertezas para oferecer uma correção de erro mais forte. Essa técnica de trabalho com a incerteza de um bit é chamada **decodificação de decisão soft**. Por outro lado, a tarefa de decidir se cada bit é 0 ou 1 antes da correção de erro subsequente é chamada **decodificação de decisão hard**.

O terceiro tipo de código de correção de erro que descreveremos é o **código de Reed-Solomon**. Assim como os códigos de Hamming, os de Reed-Solomon são códigos de bloco lineares, e eles normalmente também são sistemáticos. Diferentemente dos códigos de Hamming, que operam sobre bits individuais, os códigos de Reed-Solomon operam sobre *m* símbolos de bit. Naturalmente, a matemática é mais complicada, de modo que descreveremos sua operação por analogia.

Os códigos de Reed-Solomon são baseados no fato de que cada polinômio de grau n é determinado unicamente por $n + 1$ pontos. Por exemplo, uma linha que tem a forma $ax + b$ é determinada por dois pontos. Os pontos extras na mesma linha são redundantes, o que é útil para a correção de erro. Imagine que temos dois pontos de dados que representam uma linha e enviamos esses dois pontos de dados mais dois pontos de verificação escolhidos para que se encontrem na mesma linha. Se um dos pontos for recebido com erro, ainda podemos recuperar os pontos de dados passando uma linha pelos pontos recebidos. Três dos pontos estarão na linha e um ponto, aquele com erro, não estará. Encontrando a linha, corrigiremos o erro.

Os códigos de Reed-Solomon na realidade são definidos como polinômios que operam por campos finitos, mas funcionam de uma maneira semelhante. Para símbolos de m bits, as palavras de código possuem $2^m - 1$ símbolos de comprimento. Uma escolha popular é tornar $m = 8$, de modo que os símbolos são bytes. Uma palavra de código tem, então, 255 bytes de comprimento. O código (255, 233) é bastante utilizado; ele acrescenta 32 símbolos redundantes a 233 símbolos de dados. A decodificação com correção de erro é feita com um algoritmo desenvolvido por Berlekamp e Massey, que pode realizar com eficiência a tarefa de ajuste para códigos de tamanho moderado (Massey, 1969).

Os códigos de Reed-Solomon são bastante utilizados na prática, em virtude de suas fortes propriedades de correção de erro, particularmente para erros em rajada. Eles são usados para DSL, dados sobre cabo, comunicações por satélite e talvez de forma mais ubíqua em CDs, DVDs e discos Blu-ray. Por serem baseados em símbolos de *m* bits, um erro de único bit e um erro em rajada de *m* bits são tratados simplesmente como um erro de símbolo. Quando $2t$ símbolos redundantes são somados, um código de Reed-Solomon é capaz de corrigir até t erros em qualquer um dos símbolos transmitidos. Isso significa, por exemplo, que o código (255, 233), que tem 32 símbolos redundantes, pode corrigir até 16 erros de símbolo. Como os símbolos podem ser consecutivos e ter 8 bits cada um, uma rajada de erros de até 128 bits pode ser corrigida. A situação é ainda melhor se o modelo de erro for de apagamento (por exemplo, um arranhão em um CD que destrói alguns símbolos). Nesse caso, até $2t$ erros podem ser corrigidos.

Os códigos de Reed-Solomon normalmente são usados em combinação com outros códigos, como um código de convolução. O pensamento é o seguinte: os códigos de convolução são eficazes no tratamento de erros de bit isolados, mas eles falharão, provavelmente com uma rajada de erros, se houver muitos erros no fluxo de bits recebido. Acrescentando um código Reed-Solomon dentro do código de convolução, a decodificação Reed-Solomon pode liquidar as rajadas de erros, uma tarefa na qual ele é muito bom. O código completo, então, oferece boa proteção contra erros simples e em rajada.

O último código de correção de erros que analisaremos é o de **verificação de paridade com baixa densidade**, ou **LDPC** (**Low-Density Parity Check**). Os códigos LDPC são códigos de bloco lineares que foram inventados por Robert Gallagher em sua tese de doutorado (Gallagher, 1962). Assim como na maioria das teses, eles foram prontamente

esquecidos, para ser reinventados apenas em 1995, quando os avanços no poder de computação os tornaram viáveis.

Em um código LDPC, cada bit de saída é formado a partir de apenas uma fração dos bits de entrada. Isso leva a uma representação de matriz do código, que tem uma baixa densidade de 1s, daí o nome para o código. As palavras de código recebidas são decodificadas com um algoritmo de aproximação que melhora interativamente com um melhor ajuste dos dados recebidos a uma palavra de código válida. Isso corrige erros.

Códigos LDPC são úteis para grandes tamanhos de bloco e possuem excelentes capacidades de correção de erro que, na prática, superam muitos outros códigos (incluindo aqueles que já examinamos). Por esse motivo, eles estão sendo rapidamente incluídos em novos protocolos. Eles fazem parte do padrão para difusão de vídeo digital, 10 Gbps Ethernet, redes de linha de energia e a versão mais recente do 802.11. Você deverá ver muito mais deles nas redes do futuro.

3.2.2 Códigos de detecção de erros

Os códigos de correção de erros são extensamente utilizados em enlaces sem fios, conhecidos por ser ruidosos e propensos a erros em comparação à fiação de cobre ou à fibra óptica. Sem códigos de correção de erros, seria difícil conseguir algo. Porém, usando-se fio de cobre ou fibra de alta qualidade, a taxa de erros é muito mais baixa e, assim, a detecção de erros e a retransmissão em geral são mais eficientes para lidar com o erro ocasional.

Examinaremos três códigos de detecção de erros. Todos eles são códigos de bloco lineares e sistemáticos:

1. paridade;
2. checksums;
3. verificações de redundância cíclica (CRCs).

Para ver como eles podem ser mais eficientes do que os códigos de correção de erros, considere o primeiro código de detecção de erros, em que um único **bit de paridade** é acrescentado aos dados. O bit de paridade é escolhido de modo que o número de bits 1 na palavra de código seja par (ou ímpar). Fazer isso é equivalente a calcular o bit de paridade (par) como a soma de módulo 2 ou a operação XOR dos bits de dados. Por exemplo, quando 1011010 é enviado na paridade par, um bit é acrescentado ao final, para torná-lo 10110100. Com a paridade ímpar, 1011010 torna-se 10110101. Um código com um único bit de paridade tem uma distância de 2, pois qualquer erro de único bit produz uma palavra de código com a paridade errada. Isso significa que ele pode detectar erros de único bit.

Considere um canal no qual os erros são isolados e a taxa de erros é de 10^{-6} por bit. Essa pode parecer uma taxa de erro pequena, mas é no mínimo uma taxa justa para um cabo longo que esteja desafiando a detecção de erros. Os enlaces de LAN comuns oferecem taxas de erro de bit de 10^{-10}. Defina o tamanho do bloco como 1.000 bits. Para proporcionar a correção de erros de blocos de 1.000 bits, sabemos, pela Equação 3.1, que são necessários 10 bits de verificação. Assim, um megabit de dados necessitaria de 10.000 bits de verificação. Para simplesmente detectar um bloco com um único erro de 1 bit, um bit de paridade por bloco seria suficiente. A cada 1.000 blocos, será descoberto que um bloco possui erro e um bloco extra (1.001 bits) terá de ser transmitido para reparar o erro. O overhead total para o método de detecção de erros e retransmissão é de apenas 2.001 bits por megabit de dados, contra 10.000 bits para um código de Hamming.

Uma dificuldade com esse esquema é que um único bit de paridade só pode detectar, de maneira confiável, um erro de único bit no bloco. Se o bloco tiver um erro em rajada longo, a probabilidade de que o erro seja detectado é de apenas 0,5, o que não é muito aceitável. As disparidades poderão ser consideravelmente melhoradas se cada bloco for enviado como uma matriz retangular com n bits de largura e k bits de altura. Agora, se calcularmos e enviarmos um bit de paridade para cada linha, até k erros de bit serão confiantemente detectados, desde que haja no máximo um erro por linha.

Contudo, há outra coisa que podemos fazer que oferece melhor proteção contra erros em rajada: podemos calcular os bits de paridade sobre os dados em uma ordem diferente daquela em que os bits de dados são transmitidos. Isso é chamado de **entrelaçamento**. Nesse caso, calcularemos um bit de paridade para cada uma das n colunas e enviaremos todos os bits de dados como k linhas, enviando as linhas de cima para baixo e os bits em cada linha da esquerda para a direita, da maneira normal. Na última linha, enviamos os n bits de paridade. Essa ordem de transmissão pode ser vista na Figura 3.8 para $n = 7$ e $k = 7$.

O entrelaçamento é uma técnica geral para converter um código que detecta (ou corrige) erros isolados em um código que detecta (ou corrige) erros em rajada. Na Figura 3.8, quando ocorre um erro em rajada de tamanho $n = 7$, os bits que contêm erro estão espalhados por diferentes colunas. (Um erro em rajada não implica que todos os bits estejam errados, mas sim que pelo menos o primeiro e o último estão. Na Figura 3.8, 4 bits foram invertidos por uma faixa de 7 bits.) No máximo 1 bit em cada uma das n colunas será afetado, de modo que os bits de paridade nessas colunas detectarão o erro. Esse método usa n bits de paridade sobre blocos de kn bits de dados para detectar um único erro em rajada de comprimento n ou menor.

Contudo, uma rajada de comprimento $n + 1$ passará sem ser detectada se o primeiro e o último bits forem invertidos e todos os outros bits estiverem corretos. Se o bloco estiver bastante alterado por uma extensa rajada ou por várias rajadas mais curtas, a probabilidade de que qualquer

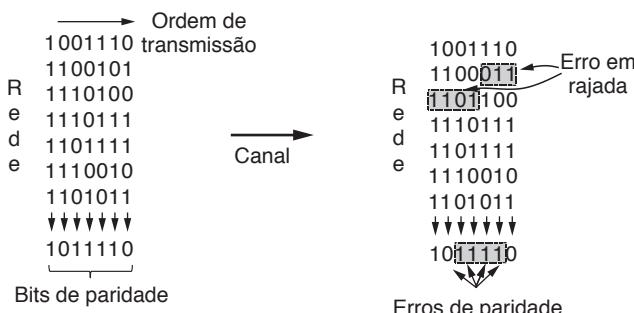



Figura 3.8 | Entrelaçamento de bits de paridade para detectar um erro em rajada.

uma das n colunas tenha a paridade correta por acidente é 0,5, de modo que a probabilidade de um bloco com problema ser aceito quando não deveria é de 2^{-n}.

O segundo tipo de código de correção de erro, o **checksum**, está bastante relacionado aos grupos de bits de paridade. O termo "checksum" normalmente é usado para indicar um grupo de bits de verificação associados a uma mensagem, independentemente de como são calculados. Um grupo de bits de paridade é um exemplo de checksum. Contudo, existem outros checksums, mais robustos, baseados na soma acumulada dos bits de dados da mensagem. O checksum normalmente é colocado no final da mensagem, como complemento da função de soma. Desse modo, os erros podem ser detectados somando a palavra inteira de código recebida, tanto bits de dados quanto a soma de verificação. Se o resultado for zero, nenhum erro foi detectado.

Um exemplo de checksum é a soma de verificação de 16 bits da Internet usada em todos os pacotes da rede como parte do protocolo IP (Braden et al., 1988). Esse checksum é uma soma dos bits da mensagem dividida por palavras de 16 bits. Como esse método opera sobre palavras, em vez de bits, assim como na paridade, os erros que deixam a paridade inalterada ainda podem alterar a soma e ser detectados. Por exemplo, se o bit de mais baixa ordem em duas palavras for invertido de 0 para 1, uma verificação de paridade entre esses bits deixaria de detectar um erro. Contudo, dois 1s serão acrescentados ao checksum de 16 bits para produzir um resultado diferente. O erro pode, então, ser detectado.

O checksum da Internet é calculado com a aritmética de complemento de um, em vez da soma de módulo 2^{16}. Na aritmética de complemento de um, um número negativo é o complemento bit a bit de seu correspondente positivo. Os computadores modernos trabalham na aritmética de complemento de dois, em que um número negativo é o complemento de um mais um. Em um computador com complemento de dois, a soma no complemento de um é equivalente a apanhar o total em módulo 2^{16} e somar qualquer overflow dos bits de alta ordem aos de baixa ordem. Esse algoritmo oferece uma cobertura mais uniforme dos dados pelos bits do checksum. De outra forma, dois bits de alta ordem podem ser somados, gerar overflow e ser perdidos sem alterar o total. Também existe outro benefício. O complemento de um tem duas representações de zero, todos os bits 0 e todos os bits 1. Isso permite um valor (por exemplo, todos os bits 0) para indicar que não há um checksum, sem a necessidade de outro campo.

Ao longo de décadas, sempre foi considerado que os quadros a ser somados para verificação contêm bits aleatórios. Todas as análises dos algoritmos de checksum têm sido feitas sob essa hipótese. A inspeção de dados reais por Partridge et al. (1995) mostrou que essa suposição é bastante errada. Como consequência, os erros não detectados são, em alguns casos, muito mais comuns do que se havia imaginado.

O checksum da Internet em particular é eficiente e simples, mas oferece uma proteção fraca em alguns casos, exatamente porque essa é uma soma simples. Ele não detecta a exclusão ou o acréscimo de dados zero, nem a troca de partes da mensagem, e oferece pouca proteção contra pedaços da mensagem em que partes de dois pacotes são reunidas. A ocorrência desses erros pode parecer improvável por processos aleatórios, mas eles são simplesmente o tipo de erro que pode ocorrer com um hardware defeituoso.

Uma escolha melhor é o **checksum de Fletcher** (Fletcher, 1982). Ele inclui um componente posicional, somando o produto dos dados e sua posição à soma acumulada. Isso oferece melhor detecção das mudanças na posição dos dados.

Embora os dois esquemas anteriores às vezes possam ser adequados em camadas superiores, na prática, um terceiro tipo mais forte de código de detecção de erro tem o uso generalizado na camada de enlace: é o **código de redundância cíclica**, ou **CRC** (**Cyclic Redundancy Check**), também conhecido como **código polinomial**. Os códigos polinomiais são baseados no tratamento de sequências de bits como representações de polinômios com coeficientes de 0 e 1 apenas. Um quadro de k bits é considerado como a lista de coeficientes para um polinômio com k termos, variando de x^{k-1} a x^0. Dizemos que tal polinômio é de grau $k - 1$. O bit de alta ordem (mais à esquerda) é o coeficiente de x^{k-1}, o próximo bit é o coeficiente de x^{k-2}, e assim por diante. Por exemplo, 110001 tem 6 bits e, portanto, representa um polinômio de seis termos com coeficientes 1, 1, 0, 0, 0 e 1: $1x^5 + 1x^4 + 0x^3 + 0x^2 + 0x^1 + 1x^0$.

A aritmética de polinômios é feita em módulo 2, de acordo com as regras da teoria algébrica. Ela não tem 'vai uns' para a adição ou empréstimos para a subtração. Tanto a adição quanto a subtração são idênticos ao XOR. Por exemplo:

$$
\begin{array}{r} 10011011 \\ +\ \underline{11001010} \\ 01010001 \end{array}
\qquad
\begin{array}{r} 00110011 \\ +\ \underline{11001101} \\ 11111110 \end{array}
\qquad
\begin{array}{r} 11110000 \\ -\ \underline{10100110} \\ 01010110 \end{array}
\qquad
\begin{array}{r} 01010101 \\ -\ \underline{10101111} \\ 11111010 \end{array}
$$

A divisão longa é efetuada do mesmo modo que em binário, exceto pelo fato de a subtração ser de módulo 2, como mostramos anteriormente. Diz-se que um divisor 'cabe em' um dividendo se o dividendo tem a mesma quantidade de bits do divisor.

Quando o método do código polinomial é empregado, o transmissor e o receptor devem concordar em relação a um **polinômio gerador**, $G(x)$, antecipadamente. Tanto o bit de mais alta ordem quanto o de mais baixa ordem do polinômio gerador devem ser iguais a 1. Para calcular o CRC de um quadro com m bits, que corresponde ao polinômio $M(x)$, o quadro deve ter mais bits do que o polinômio gerador. A ideia é acrescentar um CRC ao final do quadro, de forma que o polinômio representado pelo quadro verificado pela soma seja divisível por $G(x)$. Quando obtiver o quadro verificado, o receptor tentará dividi-lo por $G(x)$. A existência de resto indica que houve um erro de transmissão.

O algoritmo para calcular o CRC é o seguinte:

1. Seja r o grau de $G(x)$. Acrescente r bits zero à extremidade de baixa ordem do quadro, de modo que ele passe a conter $m + r$ bits e corresponda ao polinômio $x^rM(x)$.
2. Divida a string de bits correspondente a $G(x)$ pela string de bits correspondente a $x^rM(x)$ utilizando a divisão de módulo 2.
3. Subtraia o resto (que tem sempre r ou menos bits) da string de bits correspondente a $x^rM(x)$ utilizando a subtração de módulo 2. O resultado é o quadro verificado pela soma que deverá ser transmitido. Chame o polinômio de $T(x)$.

A Figura 3.9 ilustra o cálculo referente a um quadro 1101011111, usando o gerador $G(x) = x^4+x+1$.

Deve ficar claro que $T(x)$ é divisível (em módulo 2) por $G(x)$. Em qualquer problema de divisão, se você subtrair o resto do dividendo, o resultado será divisível pelo divisor. Por exemplo, na base 10, se você dividir 210.278 por 10.941, o resto será 2.399. Subtraindo-se 2.399 de 210.278, o resultado final (207.879) será divisível por 10.941.

Agora vamos analisar o poder desse método. Que tipo de erro será detectado? Imagine que ocorra um erro de transmissão de forma que, em lugar de chegar a sequência de bits correspondente a $T(x)$, seja recebida a soma $T(x) + E(x)$. Cada bit 1 em $E(x)$ corresponde a um bit que foi invertido. Se houver k bits 1 em $E(x)$, isso significa que ocorreram k erros de bits simples. Um único erro em rajada é caracterizado por um bit 1 inicial, uma mistura de bits 0 e 1 e um bit 1 final, sendo todos os outros bits iguais a 0.

Ao receber o quadro com o checksum, o receptor o divide por $G(x)$; ou seja, ele calcula $[T(x) + E(x)]/G(x)$. $T(x)/G(x)$ é igual a 0; portanto, o resultado do cálculo é simplesmente $E(x)/G(x)$. Os erros que corresponderem a polinômios contendo $G(x)$ como fator serão simplesmente ignorados; todos os outros serão descobertos.

Se houver ocorrido um erro de único bit, $E(x) = x^i$, onde i determina o bit incorreto. Se contiver dois ou mais termos, $G(x)$ nunca dividirá $E(x)$; portanto, todos os erros de único bit serão detectados.

Se tiverem ocorrido dois erros isolados de único bit, $E(x) = x^i + x^j$, onde $i > j$. Como alternativa, esse cálculo

Figura 3.9 | Exemplo de cálculo do CRC.

pode ser representado como $E(x) = x^j(x^{i-j} + 1)$. Se considerarmos que $G(x)$ não é divisível por x, uma condição suficiente para todos os erros duplos serem detectados é que $G(x)$ não divida $x^k + 1$ para qualquer k até o valor máximo de $i - j$ (isto é, até o comprimento máximo do quadro). São conhecidos polinômios simples, de grau baixo, que protegem quadros longos. Por exemplo, $x^{15} + x^{14} + 1$ não dividirá $x^k + 1$ para qualquer valor de k abaixo de 32.768.

Se houver um número ímpar de bits com erros, $E(x)$ conterá um número ímpar de termos (por exemplo, $x^5 + x^2 + 1$, mas não $x^2 + 1$). É interessante observar que nenhum polinômio com um número ímpar de termos tem $x + 1$ como fator no sistema de módulo 2. Ao tornar $x + 1$ um fator de $G(x)$, podemos detectar todos os erros que consistem em um número ímpar de bits invertidos.

Por último, e muito importante, um código polinomial com r bits de verificação detectará todos os erros em rajada que tiverem um tamanho $\leq r$. Um erro em rajada de tamanho k pode ser representado por $x^i(x^{k-1} + ... + 1)$, onde i determina a distância entre a rajada e a extremidade direita do quadro recebido. Se contiver um termo x^0, $G(x)$ não terá x^i como fator; portanto, se o grau da expressão entre parênteses for menor que o grau de $G(x)$, o resto nunca poderá ser igual a zero.

Se o tamanho da rajada for $r + 1$, o restante da divisão por $G(x)$ será zero se, e somente se, a rajada for idêntica a $G(x)$. Por definição de rajada, o primeiro e o último bits devem ser iguais a 1; assim, a correspondência entre os valores dependerá dos $r - 1$ bits intermediários. Se todas as combinações forem consideradas igualmente prováveis, a probabilidade de esse quadro incorreto ser aceito como válido será de $½^{r-1}$.

Também podemos mostrar que, ao ocorrer um erro em rajada com mais de $r + 1$ bits ou forem registradas várias rajadas mais curtas, a probabilidade de um quadro defeituoso passar despercebido poderá ser igual a $½^r$, supondo-se que todos os padrões de bits sejam igualmente prováveis.

Certos polinômios se tornaram padrões internacionais. O que é utilizado no IEEE 802 acompanhou o exemplo da Ethernet e é:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$$

Entre outras características interessantes, ele tem a propriedade de detectar todas as rajadas de comprimento 32 ou menores e todas as rajadas que afetam um número ímpar de bits. Ele tem sido muito usado desde a década de 1980. Contudo, isso não significa que seja a melhor escolha. Usando uma busca computacional completa, Castagnoli et al. (1993) e Koopman (2002) encontraram os melhores CRCs. Esses CRCs têm uma distância de Hamming de 6 para os tamanhos de mensagem típicos, enquanto o padrão do IEEE CRC-32 tem uma distância de Hamming de apenas 4.

Apesar de o cálculo necessário para computar o checksum poder parecer complicado, Peterson e Brown (1961) mostraram que é possível criar um simples circuito shift register (registrador de deslocamento) para calcular e conferir os CRCs no hardware. Na prática, esse hardware quase sempre é utilizado. Dezenas de padrões de rede compreendem diversos CRCs, incluindo praticamente todas as LANs (por exemplo, Ethernet, 802.11) e enlaces ponto a ponto (por exemplo, pacotes sobre SONET).

3.3 Protocolos básicos de enlace de dados

Como uma introdução ao estudo dos protocolos, vamos começar examinando três protocolos com graus de complexidade crescentes. Para os leitores interessados, há um simulador disponível para esses e outros protocolos na Web (veja o Prefácio). Antes de examinarmos os protocolos, é útil esclarecer algumas das suposições nas quais se baseia o modelo de comunicação.

Para começar, supomos que, na camada física, na camada de enlace de dados e na camada de rede existem processos independentes que se comunicam pelo envio de mensagens. Uma implementação comum aparece na Figura 3.10. O processo da camada física e parte do processo da camada de enlace de dados funcionam em hardware dedicado, chamado **placa de interface de rede**, ou **NIC** (**Network Interface Card**). O restante do processo da camada de enlace e o processo da camada de rede atuam sobre a CPU principal como parte do sistema operacional, com o software para o processo da camada de enlace normalmente tomando a forma de um **driver de dispositivo**. No entanto, outras implementações também são possíveis (por exemplo, três processos transferidos para um hardware dedicado, chamado **acelerador de rede**, ou três processos rodando na CPU principal a uma razão definida pelo software). Na realidade, a implementação preferida muda de uma década para a outra, com as mudanças de tecnologia. De qualquer forma, tratar as três camadas como processos separados torna a discussão conceitualmente mais clara e também enfatiza a independência das camadas.

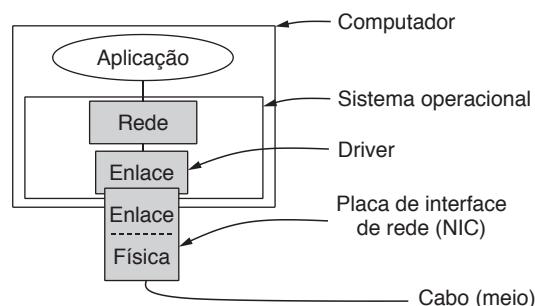



Figura 3.10 | Implementação das camadas física, de enlace de dados e de rede.

Outra suposição de extrema importância é de que a máquina *A* deseja enviar um longo fluxo de dados à máquina *B* utilizando um serviço confiável e orientado a conexões. Mais adiante, consideraremos a situação em que *B* também deseja enviar dados a *A* simultaneamente. Supõe-se que *A* tenha um suprimento infinito de dados prontos para ser enviados, e nunca terá de esperar até que eles sejam produzidos. Quando a camada de dados de *A* solicitar dados, a camada de rede sempre será capaz de obedecer de imediato. (Mais adiante essa restrição também será superada.)

Também supomos que as máquinas não sofrerão panes. Isto é, esses protocolos lidam com erros de comunicação, mas não com os problemas causados por computadores que sofrem panes e são reiniciados.

No que se refere à camada de enlace de dados, o pacote repassado a ela pela camada de rede através da interface consiste em dados puros, em que cada bit deve ser entregue à camada de rede de destino. O fato de a camada de rede de destino interpretar parte do pacote como um cabeçalho não tem nenhum interesse para a camada de enlace de dados.

Quando a camada de enlace de dados aceita um pacote, ela o encapsula em um quadro, acrescentando-lhe um cabeçalho e um final de enlace de dados (veja a Figura 3.1). Portanto, um quadro consiste em um pacote incorporado em algumas informações de controle (no cabeçalho) e em um checksum (no final). Em seguida, o quadro é transmitido à camada de enlace de dados da outra máquina. Presumiremos que existem funções de biblioteca adequadas, *to_physical_layer* para enviar um quadro e *from_physical_layer* para receber um quadro. Essas funções calculam ou acrescentam o checksum (o que normalmente é feito no hardware), de forma que os protocolos que desenvolvemos nesta seção não precisam se preocupar com isso. Por exemplo, eles poderiam usar o algoritmo de CRC discutido na seção anterior.

Inicialmente, o receptor nada tem a fazer. Ele apenas fica à espera de que algo aconteça. Nos exemplos de protocolos apresentados neste capítulo, indicaremos que a camada de enlace de dados está esperando que algo aconteça por meio da chamada da função *wait_for_event*(&*event*). Essa função só retorna quando acontece algo (por exemplo, quando chega um quadro). Ao retornar, a variável *event* informa o que aconteceu. O conjunto de eventos possíveis é diferente para os diversos protocolos a ser descritos e será definido separadamente para cada protocolo. Observe que, em uma situação mais realista, a camada de enlace de dados não ficará em um loop estrito à espera de um evento, como sugerimos, mas receberá uma interrupção, o que a fará interromper o que quer que esteja fazendo para manipular o quadro recebido. Apesar disso, por simplicidade, ignoraremos todos os detalhes de atividades paralelas na camada de enlace de dados, e presumiremos que ela se dedica em tempo integral apenas ao tratamento do nosso canal.

Quando um quadro chega ao receptor, o checksum é recalculado. Se o checksum no quadro estiver incorreto (ou seja, se houve um erro de transmissão), a camada de enlace de dados será informada (*event* = *cksum_err*). Se o quadro recebido tiver chegado intacto, a camada de enlace de dados também será informada (*event* = *frame_arrival*), para que ela possa receber o quadro para inspeção usando *from_physical_layer*. Assim que recebe um quadro sem danos, a camada de enlace de dados verifica as informações de controle contidas no cabeçalho e, se tudo estiver correto, repassa a porção do pacote à camada de rede. Em nenhuma circunstância o cabeçalho do quadro será entregue à camada de rede.

Há uma boa razão para que a camada de rede nunca receba nenhuma parte do cabeçalho do quadro: manter os protocolos de rede e de enlace de dados completamente separados. Desde que a camada de rede não saiba absolutamente nada sobre o protocolo de enlace de dados ou sobre o formato do quadro, esses itens poderão ser alterados sem exigir mudanças no software da camada de rede. Isso acontece sempre que uma nova NIC é instalada em um computador. A utilização de uma interface rígida entre a camada de rede e a de enlace de dados simplifica bastante o projeto do software, pois os protocolos de comunicação das diferentes camadas podem evoluir de forma independente.

O Quadro 3.1 mostra algumas declarações (na linguagem C) comuns a muitos dos protocolos que serão discutidos mais adiante. Cinco estruturas de dados são definidas nesse quadro: *boolean*, *seq_nr*, *packet*, *frame_kind* e *frame*. Um *boolean* é do tipo enumerado e pode assumir os valores verdadeiro (*true*) e falso (*false*). Um *seq_nr* é um inteiro pequeno usado para numerar os quadros, para facilitar sua distinção. Esses números de sequência variam de 0 até *MAX_SEQ* (inclusive), que representa um limite a ser definido, quando necessário, para cada protocolo. Um *packet* é a unidade de informação trocada entre as camadas de rede e de enlace de dados da mesma máquina, ou entre pares da camada de rede. Em nosso modelo, ele sempre contém *MAX_PKT* bytes; no entanto, de modo mais realista, ele teria comprimento variável.

Um *quadro* é composto de quatro campos: *kind*, *seq*, *ack* e *info*; os três primeiros contêm informações de controle, e o último pode conter os dados reais a ser transferidos. Esses campos de controle são chamados coletivamente **cabeçalho do quadro**.

O campo *kind* indica se há dados no quadro, pois alguns protocolos distinguem quadros que contêm exclusivamente informações de controle daqueles que armazenam dados além dessas informações. Os campos *seq* e *ack* são usados para números de sequência e confirmações, respectivamente; seu uso será descrito detalhadamente mais adiante. O campo *info* de um quadro de dados contém um único pacote; o campo *info* de um quadro de controle não é usado. Uma implementação mais realista utilizaria um

```
#define MAX_PKT 1024                                   /* determina tamanho do pacote em bytes */

typedef enum {false, true} boolean;                    /* tipo boolean */
typedef unsigned int seq_nr;                           /* números de sequência ou ack */
typedef struct {unsigned char data[MAX_PKT];}packet;   /* definição do pacote */
typedef enum {data, ack, nak} frame_kind;              /* definição de frame_kind */

typedef struct {                                       /* quadros são transportados nesta camada */
frame_kind kind;                                       /* que tipo de quadro é este? */
seq_nr seq;                                            /* número de sequência */
seq_nr ack;                                            /* número de confirmação */
packet info;                                           /* o pacote da camada de rede */
} frame;

/* Espera que um evento aconteça; retorna o tipo de evento em event. */
void wait_for_event(event_type*event);

/* Busca um pacote da camada de rede para transmissão pelo canal. */
void from_network_layer(packet*p);

/* Entrega informação de um quadro que chega à camada de rede. */
void to_network_layer(packet*p);

/* Recebe um quadro de entrada da camada física e o copia para r. */
void from_physical_layer(frame*r);

/* Passa o quadro à camada física para transmissão. */
void to_physical_layer(frame*s);

/* Inicia o timer e habilita o evento timeout. */
void start_timer(seq_nr k);

/* Termina o timer e desativa o evento timeout. */
void stop_timer(seq_nr k);

/* Inicia um timer auxiliar e habilita o ack_timeout event. */
void start_ack_timer(void);

/* Encerra o timer auxiliar e desabilita o evento ack_timeout. */
void stop_ack_timer(void);

/* Permite que a camada de rede gere um evento network_layer_ready. */
void enable_network_layer(void);

/* Proíbe a camada de rede de um evento network_layer_ready. */
void disable_network_layer(void);

/* Macro inc é expandido em linha: incrementa k de modo circular. */
#define inc(k) if (k < MAX_SEQ)k = k + 1; else k = 0
```

Quadro 3.1 | Algumas definições utilizadas nos protocolos apresentados a seguir. Essas definições estão armazenadas no arquivo *protocol.h*.

campo *info* de comprimento variável; nos quadros de controle, esse campo seria completamente omitido.

Novamente, é importante compreender o relacionamento entre um pacote e um quadro. A camada de rede cria um pacote tomando uma mensagem da camada de transporte e acrescentando a ela o cabeçalho da camada de rede. Esse pacote é repassado à camada de enlace de dados para inclusão no campo *info* de um quadro que esteja sendo enviado. Quando o quadro chega ao destino, a camada de enlace de dados extrai o pacote do quadro e o envia à camada

de rede. Dessa forma, a camada de rede pode atuar como se as máquinas pudessem trocar pacotes diretamente.

No Quadro 3.1 também estão listadas diversas funções. Essas funções são rotinas de biblioteca cujos detalhes são dependentes da implementação e cujo funcionamento interno não será discutido aqui. A função *wait_for_event* permanece à espera de que algo aconteça, como mencionamos anteriormente. As funções *to_network_layer* e *from_network_layer* são usadas pela camada de enlace de dados para enviar pacotes à camada de rede e aceitar pacotes dessa camada, respectivamente. Observe que *from_physical_layer* e *to_physical_layer* repassam quadros entre a camada de enlace de dados e a camada física. Em outras palavras, *to_network_layer* e *from_network_layer* lidam com a interface entre as camadas 2 e 3, enquanto *from_physical_layer* e *to_physical_layer* lidam com a interface entre as camadas 1 e 2.

Na maioria dos protocolos, supomos o uso de um canal não confiável que perde quadros inteiros ocasionalmente. Para se recuperar dessas calamidades, sempre que envia um quadro, a camada de enlace de dados transmissora tem de inicializar um timer ou timer interno. Se nenhuma confirmação tiver sido recebida dentro de um intervalo predeterminado, o timer expirará por timeout e a camada de enlace de dados receberá um sinal de interrupção.

Em nossos protocolos, isso é tratado permitindo-se à função *wait_for_event* retornar *event = timeout*. As funções *start_timer* e *stop_timer* ativam e desativam o timer, respectivamente. Os timeouts só são possíveis quando o timer está funcionando e antes que *stop_timer* seja chamado. É explicitamente permitido chamar *start_timer* enquanto o timer está funcionando; esse tipo de chamada simplesmente reinicializa o timer para provocar o próximo timeout, depois de decorrer um intervalo do timer (a menos que ele seja reiniciado ou desativado durante esse intervalo).

As funções *start_ack_timer* e *stop_ack_timer* controlam um timer auxiliar cuja finalidade é gerar confirmações sob determinadas condições.

As funções *enable_network_layer* e *disable_network_layer* são usadas nos protocolos mais sofisticados, para os quais não mais supomos que a camada de rede sempre terá pacotes a ser enviados. Quando a camada de enlace de dados habilita a camada de rede, esta passa a ter permissão para causar uma interrupção sempre que tiver um pacote para enviar. Isso é indicado por *event = network_layer_ready*. Quando a camada de rede está inativa, ela não pode causar tais eventos. Definindo com cuidado os momentos em que ativa e desativa a camada de rede, a camada de enlace de dados pode impedir que a camada de rede acabe ficando sobrecarregada com pacotes para os quais não dispõe de espaço no buffer.

Os números de sequência dos quadros estão sempre na faixa de 0 a *MAX_SEQ* (inclusive), onde *MAX_SEQ* tem um valor diferente para os diversos protocolos. Com frequência, é necessário aumentar um número de sequência em uma unidade, de forma circular (isto é, *MAX_SEQ* é seguido por 0). A macro *inc* cuida desse incremento. Ela é definida como uma macro porque é usada em linha no caminho crítico. Como veremos adiante, com frequência o processamento de protocolos é o fator que limita o desempenho da rede; portanto, a definição de operações simples como macros não afeta a legibilidade do código, mas melhora o desempenho.

As declarações do Quadro 3.1 fazem parte de cada um dos protocolos brevemente apresentados a seguir. Para economizar espaço e facilitar a consulta, essas declarações foram extraídas dos protocolos e são apresentadas juntas, mas conceitualmente elas devem estar integradas aos protocolos. Na linguagem C, essa integração é feita inserindo-se as definições em um arquivo de cabeçalho especial, neste caso *protocol.h*, e utilizando-se o recurso #include do pré-processador C, que inclui essas definições nos arquivos de protocolo.

3.3.1 Um protocolo simplex sem restrições

Como primeiro exemplo, consideraremos o protocolo mais simples possível, pois não se preocupa com a possibilidade de algo sair errado. Os dados são transmitidos em apenas um sentido. As camadas de rede do transmissor e do receptor estão sempre prontas. O tempo de processamento pode ser ignorado. O espaço disponível em buffer é infinito. E o melhor de tudo é que o canal de comunicação entre as camadas de enlace de dados nunca é danificado nem perde quadros. Esse protocolo absolutamente imaginário, que denominaremos "utopia", é mostrado no Quadro 3.2.

O protocolo consiste em dois procedimentos distintos, um que envia informações e outro que as recebe. O procedimento no transmissor é executado na camada de enlace de dados da máquina de origem, e no receptor é executado na camada de enlace de dados da máquina de destino. Não são usados números de sequência ou de confirmação; portanto, *MAX_SEQ* não é necessário. O único tipo de evento possível é *frame_arrival* (ou seja, a chegada de um quadro não danificado).

No transmissor há um loop while infinito que envia os dados o mais rápido possível. O corpo do loop é formado por três ações: buscar um pacote da (sempre prestativa) camada de rede, criar um quadro utilizando a variável *s* e transmitir o quadro ao destino. Apenas o campo *info* do quadro é usado por esse protocolo, pois os outros campos se referem ao controle de fluxo e de erros e, nesse caso, não há erros nem restrições de controle de fluxo.

O receptor é igualmente simples. No início, ele espera que algo aconteça, e a única possibilidade é a chegada de um quadro não danificado. Finalmente, o quadro chega e a função *wait_for_event* retorna, com *event* definido como *frame_arrival* (o que, de qualquer forma, é ignorado). A chamada *from_physical_layer* remove o quadro recém-chegado do buffer do hardware e o coloca na variável *r*, onde o código

receptor poderá buscá-lo quando necessário. Por fim, a parte referente aos dados é repassada à camada de rede, e a camada de enlace de dados volta a esperar pelo próximo quadro, ficando efetivamente em suspenso até a chegada do quadro.

O protocolo sem restrições é imaginário porque não trata nem do controle de fluxo nem da correção de erro. Seu processamento é próximo ao de um serviço não confirmado não orientado a conexões, que conta com as camadas mais altas para resolver esses problemas, embora até mesmo um serviço desse tipo realize alguma detecção de erros.

3.3.2 Um protocolo simplex stop-and-wait em um canal livre de erros

Agora, trataremos do problema de impedir que o transmissor sobrecarregue o receptor com quadros mais rapidamente do que ele conseguirá processá-los. Essa situação pode facilmente acontecer na prática, de modo que é muito importante poder impedi-la. No entanto, continuamos supondo que o canal de comunicação não apresenta erros e que o tráfego de dados ainda é do tipo simplex.

```
/* O protocolo 1 (utopia) oferece transmissão de dados em um único sentido, do transmissor para o receptor.
   Pressupõe-se que o canal de comunicação é livre de erros e que o receptor é capaz de processar toda
   a entrada de uma forma infinitamente rápida. Consequentemente, o transmissor permanece em um loop
   enviando os dados com a maior rapidez possível. */

typedef enum {frame_arrival}event_type;
#include "protocol.h"

void sender1(void)
{
  frame s;                              /* buffer para um quadro de saída */
  packet buffer;                        /* buffer para um pacote de saída */

  while (true) {
     from_network_layer(&buffer);       /* pega algo para enviar */
     s.info = buffer;                   /* copia para s, para transmissão */
     to_physical_layer(&s);             /* envia-o pelo caminho */
  }                                     /* O amanhã, o amanhã, o amanhã,
                                        avança em pequenos passos, dia após dia,
                                        até a última sílaba da recordação.
                                           – Macbeth, V, v */
}

void receiver1(void)
{
  frame r;
  event_type event;                     /* preenchido pela espera, mas não usado aqui */

  while (true) {
     wait_for_event(&event);            /* única possibilidade é frame_arrival*/
     from_physical_layer(&r);           /* recebe o quadro que chega */
     to_network_layer(&r.info);         /* passa os dados à camada de rede */
  }
}
```

Quadro 3.2 | Um protocolo simplex sem restrições.

Uma solução é montar o receptor para que seja poderoso o bastante para processar um fluxo contínuo de quadros de ponta a ponta (ou, de modo equivalente, definir a camada de enlace para que seja lenta o bastante para que o receptor possa acompanhar). Ela deverá ter buffer e capacidade de processamento suficientes para atuar na velocidade da linha e deve ser capaz de passar os quadros recebidos à camada de rede com rapidez suficiente. Porém, essa é uma solução no pior dos casos. Ela exige hardware dedicado e pode desperdiçar recursos se a utilização do enlace for quase sempre baixa. Além do mais, ela apenas passa o problema de lidar com um emissor muito rápido para outro lugar; nesse caso, para a camada de rede.

Uma solução mais geral para esse problema é fazer com que o receptor ofereça feedback ao transmissor. Depois de enviar um pacote à sua camada de rede, o receptor envia um pequeno quadro fictício de volta ao transmissor, permitindo a transmissão do próximo quadro. Após o envio de um quadro, o protocolo exige que o transmissor espere sua vez, até a chegada de um pequeno quadro fictício (isto é, de confirmação). Esse atraso é um exemplo simples de protocolo de controle de fluxo.

Os protocolos nos quais o transmissor envia um quadro e em seguida espera por uma confirmação antes de continuar sua operação são chamados **stop-and-wait** (pare e espere). O Quadro 3.3 mostra um exemplo de protocolo simplex stop-and-wait.

Apesar de o tráfego de dados nesse exemplo ser simplex, indo apenas do transmissor ao receptor, os quadros são enviados em ambas as direções. Consequentemente, o canal de comunicação entre as duas camadas de enlace de dados deve ser capaz de realizar a transferência bidirecional

```
/* O protocolo 2 (stop-and-wait) também implementa um fluxo de dados unidirecional entre o transmissor e
   o receptor. Presume-se mais uma vez que o canal de comunicação seja totalmente livre de erros, como
   no protocolo 1. No entanto, dessa vez, o receptor tem buffer e velocidade de processamento finitos;
   portanto, o protocolo deverá impedir explicitamente que o transmissor sobrecarregue o receptor enviando
   dados mais rapidamente do que ele é capaz de processar. */

typedef enum {frame_arrival}event_type;
#include "protocol.h"

void sender2(void)
{
  frame s;                              /* buffer para um quadro de saída */
  packet buffer;                        /* buffer para um pacote de saída */
  event_type event;                     /* frame_arrival é a única possibilidade */

  while (true) {
     from_network_layer(&buffer);       /* apanha algo para enviar */
     s.info = buffer;                   /* copia para s, para transmissão */
     to_physical_layer(&s);             /* pequeno quadro de adeus */
     wait_for_event(&event);            /* não avança até um sinal verde */
  }
}

void receiver2(void)
{
  frame r, s;                           /* buffers para quadros */
  event_type event;                     /* frame_arrival é a única possibilidade */
  while (true) {
     wait_for_event(&event);            /* a única possibilidade é frame_arrival*/
     from_physical_layer(&r);           /* apanha o quadro de entrada */
     to_network_layer(&r.info);         /* passa os dados para a camada de rede */
     to_physical_layer(&s);             /* envia quadro fictício para acordar o transmissor */
  }
}
```

Quadro 3.3 | Um protocolo simplex stop-and-wait.

de informações. No entanto, esse protocolo acarreta uma rígida alternância de fluxo: primeiro o transmissor envia um quadro, depois o receptor envia outro; em seguida, o transmissor envia mais um quadro e assim por diante. Um canal físico half-duplex seria suficiente nesse caso.

A exemplo do protocolo 1, o transmissor começa extraindo um pacote da camada de rede, utilizando-o para criar um quadro que em seguida é transmitido a seu destino. Porém, agora, ao contrário do que ocorre no protocolo 1, o transmissor deve aguardar a chegada de um quadro de confirmação antes de tornar a entrar em loop e buscar o próximo pacote da camada de rede. A camada de enlace de dados do transmissor não precisa sequer inspecionar o quadro recebido, pois só há uma possibilidade: o quadro recebido é sempre uma confirmação.

A única diferença entre *receiver1* e *receiver2* é que, após entregar um pacote à camada de rede, o *receiver2* envia um quadro de confirmação de volta ao transmissor, antes de entrar mais uma vez no loop de espera. Como apenas a chegada do quadro de volta ao transmissor é importante, e não seu conteúdo, o receptor não precisa incluir nenhuma informação específica no quadro.

3.3.3 Um protocolo simplex stop-and-wait em um canal com ruído

Agora, vamos considerar a situação normal de um canal de comunicação no qual ocorrem erros. Os quadros podem ser danificados ou completamente perdidos. No entanto, supomos que, se um quadro for danificado em trânsito, o hardware receptor detectará essa ocorrência ao calcular o checksum. Se o quadro for danificado de tal forma que o checksum nunca esteja correto — uma possibilidade muito improvável —, o protocolo em questão (e todos os outros protocolos) poderá apresentar falhas (isto é, poderá entregar um pacote incorreto à camada de rede).

À primeira vista, pode parecer que uma variação do protocolo 2 seria viável com a inclusão de um timer. O transmissor poderia enviar um quadro, mas o receptor só enviaria um quadro de confirmação se os dados fossem recebidos corretamente. Se um quadro danificado chegasse ao receptor, ele seria descartado. Após certo tempo, o transmissor alcançaria seu timeout e enviaria o quadro mais uma vez. Esse processo seria repetido até que o quadro finalmente chegasse intacto.

Esse esquema tem uma falha fatal. Pense no problema e tente descobrir o que poderia estar errado antes de continuar a leitura.

Para verificar o que poderia estar errado, lembre-se de que a função dos processos da camada de enlace de dados é oferecer comunicações transparentes e livres de erros entre os processos da camada de rede. A camada de rede da máquina *A* envia uma série de pacotes à camada de enlace de dados da mesma máquina. Esta, por sua vez, deve se certificar de que a camada de enlace de dados da máquina *B* enviará uma série idêntica de pacotes à camada de rede da mesma máquina. Em particular, a camada de rede da máquina *B* não tem como saber se um pacote foi perdido ou duplicado; portanto, a camada de enlace de dados deve garantir que nenhuma combinação de erros de transmissão, mesmo improvável, possa fazer com que um pacote duplicado seja entregue à camada de rede.

Considere a seguinte situação:

1. A camada de rede de *A* envia o pacote 1 à sua camada de enlace de dados. O pacote é corretamente recebido em *B* e repassado à camada de rede de *B*. *B* envia um quadro de confirmação de volta a *A*.
2. O quadro de confirmação se perde por completo. Ele simplesmente nunca chega ao destino. Tudo seria muito mais simples se o canal tivesse adulterado e perdido apenas quadros de dados, não quadros de controle. No entanto, para nossa tristeza, o canal não faz distinção entre quadros.
3. Finalmente, a camada de enlace de dados de *A* tem seu limite de tempo esgotado. Como não recebeu uma confirmação, ela presume (incorretamente) que seu quadro de dados se perdeu ou foi danificado e envia mais uma vez o quadro contendo o pacote 1.
4. O quadro duplicado também chega perfeitamente à camada de enlace de dados de *B* e é repassado de imediato, sem maiores problemas, à sua camada de rede. Caso *A* esteja enviando um arquivo a *B*, uma parte do arquivo será duplicada (isto é, a cópia do arquivo criado por *B* estará incorreta e o erro não será detectado). Em outras palavras, o protocolo falhará.

Na verdade, precisamos dar ao receptor alguma forma de poder distinguir entre um quadro que ele está recebendo pela primeira vez e uma retransmissão. A maneira mais óbvia de conseguir isso é fazer o transmissor incluir um número de sequência no cabeçalho de cada quadro enviado. Dessa forma, o receptor poderá verificar o número de sequência de cada quadro recebido para confirmar se esse é um novo quadro ou se é uma cópia a ser descartada.

Como o protocolo deve ser correto e o campo de número de sequência no cabeçalho provavelmente é pequeno para usar o enlace de modo eficiente, surge a seguinte pergunta: qual é a quantidade mínima de bits necessária para o número de sequência? O cabeçalho poderia oferecer 1 bit, alguns bits, um byte ou múltiplos bytes para um número de sequência, dependendo do protocolo. O importante é que ele deve transportar números de sequência grandes o suficientes para que o protocolo funcione corretamente, ou o protocolo não terá valor.

A única ambiguidade nesse protocolo ocorre entre um quadro, m, e seu sucessor direto, $m + 1$. Se o quadro m tiver sido perdido ou danificado, o receptor não o confirmará; portanto, o transmissor continuará tentando enviá-lo. Uma

vez que o quadro tenha sido corretamente recebido, o receptor enviará uma confirmação de volta ao transmissor. E aqui surge um problema potencial. Dependendo do fato de o quadro de confirmação voltar ao transmissor corretamente ou não, o transmissor poderá tentar enviar *m* ou $m + 1$.

No transmissor, o evento que dispara a transmissão do quadro $m + 1$ é a chegada de uma confirmação para o quadro *m*. Mas essa situação implica que $m - 1$ foi recebido corretamente e, além disso, que sua confirmação também foi recebida corretamente pelo transmissor. Caso contrário, o transmissor não teria iniciado com *m*, muito menos estaria considerando $m + 1$. Por conseguinte, a única ambiguidade é entre um quadro e seu predecessor ou sucessor imediato, não entre os próprios predecessores ou sucessores.

Um número de sequência de 1 bit (0 ou 1) é, portanto, suficiente. A cada instante, o receptor espera o próximo número de sequência. Quando chega um quadro contendo um número de sequência correto, ele é aceito e repassado à camada de rede, e depois confirmado. Em seguida, o número de sequência esperado é incrementado na base 2 (ou seja, 0 passa a ser 1 e 1 passa a ser 0). Qualquer quadro recebido que contenha o número de sequência errado será rejeitado por ser considerado uma cópia. Porém, a última confirmação válida é repetida, de forma que o transmissor finalmente possa descobrir que o quadro foi recebido.

Um exemplo desse tipo de protocolo é mostrado no Quadro 3.4. Os protocolos nos quais o transmissor espera por uma confirmação positiva antes de passar para o próximo item de dados frequentemente são chamados **solicitação de repetição automática**, ou **ARQ** (**Automatic Repeat reQuest**), ou **confirmação positiva com retransmissão**, ou **PAR** (**Positive Acknowledgement with Retransmission**). A exemplo do protocolo 2, ele também transmite dados em apenas um sentido.

O protocolo 3 difere de seus predecessores pelo fato de tanto o transmissor quanto o receptor terem uma variável cujo valor é memorizado enquanto a camada de enlace de dados se encontra em estado de espera. Em *next_frame_to_send*, o transmissor memoriza o número de sequência do próximo quadro a ser enviado; em *frame_expected*, o receptor memoriza o número de sequência do próximo quadro esperado. Cada protocolo tem uma breve fase de inicialização antes de entrar no loop infinito.

Após enviar um quadro, o transmissor ativa o timer. Caso já esteja ativado, ele será reiniciado para permitir a contagem de outro intervalo. O intervalo deve ser escolhido de forma que haja tempo suficiente para o quadro chegar ao receptor, para o receptor processá-lo na pior das hipóteses e para o quadro de confirmação ser enviado de volta ao transmissor. Somente quando o intervalo tiver se esgotado, poderemos supor com segurança que o quadro transmitido ou sua confirmação se perdeu, e que será necessário enviar uma cópia. Se o intervalo de timeout for definido com um valor curto demais, o transmissor enviará quadros desnecessários. Embora não afetem a exatidão do protocolo, esses quadros extras prejudicarão o desempenho.

Depois de transmitir um quadro e ativar o timer, o transmissor espera que algo interessante aconteça. Existem apenas três possibilidades: o quadro de confirmação chegar sem danos, o quadro de confirmação chegar com erro ou o timer expirar. Se uma confirmação válida for recebida, o transmissor buscará o próximo pacote em sua camada de rede e o colocará no buffer, substituindo o pacote anterior. Ele também aumentará o número de sequência. Se for recebido um quadro com erro ou se o timer expirar, o buffer e o número de sequência permanecerão inalterados, de modo que uma cópia do quadro poderá ser enviada. De qualquer forma, o conteúdo do buffer (tanto o próximo pacote como uma cópia) é enviado em seguida.

Quando um quadro válido chega ao receptor, seu número de sequência é conferido, para verificar se ele é uma cópia. Se não for, o quadro será aceito, enviado à camada de rede, e uma confirmação será gerada. Cópias e quadros danificados não serão repassados à camada de rede, mas eles fazem com que o último quadro recebido corretamente seja confirmado para sinalizar ao transmissor para avançar ao próximo quadro ou retransmitir um quadro danificado.

3.4 Protocolos de janela deslizante

Nos protocolos apresentados anteriormente, os quadros de dados eram transmitidos em apenas um sentido. Em situações mais práticas, há necessidade de transmitir dados em ambos os sentidos. Você pode obter uma transmissão de dados full-duplex definindo dois canais de comunicação distintos e cada um deles usando um enlace separado para um tráfego de dados simplex (em diferentes sentidos). Cada enlace é composto de um canal 'direto' (para dados) e de um canal 'reverso' (para confirmações). Em ambos os casos, a capacidade do canal reverso é quase totalmente perdida.

Uma ideia melhor é usar o mesmo circuito para dados em ambos os sentidos. Afinal de contas, nos protocolos 2 e 3 ele já estava sendo usado para transmitir quadros em ambos os sentidos, e o canal reverso normalmente tem a mesma capacidade do canal direto. Nesse modelo, os quadros de dados enviados de *A* para *B* são misturados com os quadros de confirmação enviados de *A* para *B*. Ao verificar o campo *kind* no cabeçalho de um quadro recebido, o receptor pode identificar se o quadro é de dados ou de confirmação.

Apesar de o entrelaçamento de quadros de dados e de controle no mesmo circuito representar um grande avanço em relação ao uso de dois circuitos físicos separados, ainda é possível introduzir mais um aperfeiçoamento. Quando um quadro de dados chega a seu destino, em vez de enviar imediatamente um quadro de controle separado, o receptor se contém e espera até a camada de rede enviar o pró-

```
/* O protocolo 3 (PAR) permite que dados unidirecionais fluam por um canal não confiável. */

#define MAX_SEQ 1                             /* deve ser 1 para o protocolo 3 */
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"

void sender3(void)
{
  seq_nr next_frame_to_send;                  /* número seq do próximo quadro de saída */
  frame s;                                    /* variável auxiliar */
  packet buffer;                              /* buffer para pacote de saída */
  event_type event;

  next_frame_to_send = 0;                     /* inicia números de sequência de saída */
  from_network_layer(&buffer);                /* busca primeiro pacote */
  while (true) {
    s.info = buffer;                          /* monta um quadro para transmissão */
    s.seq = next_frame_to_send;               /* insere número de sequência no quadro */
    to_physical_layer(&s);                    /* envia o quadro */
    start_timer(s.seq);                       /* se a resposta levar muito tempo, timeout */
    wait_for_event(&event);                   /* frame_arrival, cksum_err, timeout */
    if (event == frame_arrival){
      from_physical_layer(&s);                /* obtém a confirmação */
      if (s.ack == next_frame_to_send){
        stop_timer(s.ack);                    /* desliga o timer */
        from_network_layer(&buffer);          /* pega o próximo quadro a enviar */
        inc(next_frame_to_send);              /* inverte next_frame_to_send */
      }
    }
  }
}

void receiver3(void)
{
  seq_nr frame_expected;
  frame r, s;
  event_type event;

  frame_expected = 0;
  while (true) {
    wait_for_event(&event);                   /* possibilidades: frame_arrival, cksum_err */
    if (event == frame_arrival){              /* chegou um quadro válido */
      from_physical_layer(&r);                /* pega quadro recém-chegado */
      if (r.seq == frame_expected){           /* é isso que estávamos esperando */
        to_network_layer(&r.info);            /* passa os dados para a camada de rede */
        inc(frame_expected);                  /* da próx. vez, espera outro núm. de sequência nr*/
      }
      s.ack = 1 – frame_expected;             /* diz qual quadro está sendo confirmado */
      to_physical_layer(&s);                  /* envia confirmação */
    }
  }
}
```

Quadro 3.4 | Uma confirmação positiva com protocolo de retransmissão.

ximo pacote. A confirmação é acrescentada ao quadro de dados que está sendo enviado (por meio do campo *ack* do cabeçalho do quadro). Na verdade, a confirmação pega carona no próximo quadro de dados que estiver sendo enviado. A técnica de retardar temporariamente as confirmações e enviá-las com o próximo quadro de dados é conhecida pelo nome de **piggybacking** (**pegar carona**).

A principal vantagem do piggybacking em relação ao envio de quadros de confirmação distintos é a melhor utilização da largura de banda disponível para o canal. O campo *ack* do cabeçalho do quadro precisa de apenas alguns bits, enquanto um quadro separado precisaria do cabeçalho, da confirmação e do checksum. Além disso, um número menor de quadros enviados significa uma carga de processamento menor no receptor. No próximo protocolo a ser examinado, o campo de piggyback necessita apenas de um bit no cabeçalho do quadro. Em geral, ele raramente precisa de mais que alguns bits.

No entanto, o piggybacking introduz uma complicação não presente em confirmações separadas. Quanto tempo a camada de enlace de dados deve esperar por um pacote ao qual deverá acrescentar a confirmação? Se a camada de enlace de dados esperar durante um intervalo maior que o permitido pelo timeout do transmissor, o quadro será retransmitido, o que invalidará todo o processo de confirmação. Se a camada de enlace de dados fosse um oráculo e pudesse prever o futuro, ela saberia quando o próximo pacote da camada de rede estivesse chegando e poderia decidir entre esperar por ele e enviar imediatamente uma confirmação separada, dependendo da duração prevista do tempo de espera. É óbvio que a camada de enlace de dados não é capaz de prever o futuro; portanto, ela deve recorrer a algum esquema ad hoc, como esperar durante um número fixo de milissegundos. Se um novo pacote chegar rapidamente, a confirmação será acrescentada a ele; caso contrário, se nenhum pacote tiver chegado até o final desse intervalo, a camada de enlace de dados simplesmente enviará um quadro de confirmação separado.

Os três protocolos seguintes são bidirecionais e pertencem a uma classe identificada como protocolos de **janela deslizante**. Os três apresentam diferenças em termos de eficiência, complexidade e requisitos de buffer, como discutiremos adiante. Neles, como em todos os protocolos de janela deslizante, cada quadro enviado contém um número de sequência, variando de 0 até algum valor máximo. Em geral, o valor máximo é $2^n - 1$, de forma que o número de sequência caiba exatamente em um campo de n bits. O protocolo de janela deslizante stop-and-wait utiliza $n = 1$, restringindo os números de sequência a 0 e 1; no entanto, versões mais sofisticadas podem usar um valor arbitrário de n.

A essência de todos os protocolos de janela deslizante é o fato de que, em qualquer instante, o transmissor mantém um conjunto de números de sequência correspondentes a quadros que ele pode enviar. Dizemos que esses quadros estão reunidos na **janela de transmissão**. Da mesma forma, o receptor mantém uma **janela de recepção** correspondente ao conjunto de quadros que está apto a aceitar. As janelas do transmissor e do receptor não precisam ter os mesmos limites superior e inferior ou o mesmo tamanho. Em alguns protocolos, essas janelas têm tamanho fixo, mas em outros elas podem aumentar ou diminuir à medida que os quadros são enviados e recebidos.

Apesar de esses protocolos permitirem que a camada de enlace de dados tenha mais liberdade em relação à ordem em que poderá enviar e receber quadros, definitivamente não descartamos a exigência de o protocolo entregar os pacotes à camada de rede na mesma ordem em que eles foram repassados à camada de enlace de dados na máquina transmissora. Outra exigência que não mudou é que o canal de comunicação física seja "como nos fios", ou seja, que entregue todos os quadros na ordem em que eles são enviados.

Os números de sequência contidos na janela do transmissor representam quadros que foram ou que podem ser enviados, mas ainda não confirmados. Sempre que chega um novo pacote da camada de rede, ele recebe o próximo número de sequência mais alto, e o limite superior da janela é incrementado em uma unidade. Quando uma confirmação é recebida, o limite inferior é incrementado em uma unidade. Dessa forma, a janela mantém continuamente uma lista de quadros não confirmados. A Figura 3.11 mostra um exemplo.

Tendo em vista que os quadros presentes atualmente na janela do transmissor podem ser perdidos ou danificados em trânsito, o transmissor deve manter todos esses quadros em sua memória para que a retransmissão seja possível. Assim, se o tamanho máximo da janela for n, o transmissor precisará de n buffers para armazenar os quadros não confirmados. Se a janela chegar a seu tamanho máximo, a camada de enlace de dados do transmissor será obrigada a desativar a camada de enlace de dados até que outro buffer esteja livre.

O tamanho da janela da camada de enlace de dados receptora corresponde aos quadros que ela é capaz de aceitar. Qualquer quadro que ficar fora da janela será simplesmente descartado. Quando for recebido um quadro cujo número de sequência for igual ao limite inferior da janela, ele será repassado à camada de rede, será gerada uma confirmação e a janela será incrementada em uma unidade. Qualquer quadro fora da janela é descartado. Em todos esses casos, uma confirmação subsequente é gerada para que o transmissor possa descobrir como proceder. Observe que um tamanho de janela igual a 1 significa que a camada de enlace de dados só aceita quadros em ordem, mas para janelas maiores isso não é verdade. A camada de rede, ao contrário, sempre recebe dados na ordem adequada, independentemente do tamanho da janela da camada de enlace de dados.

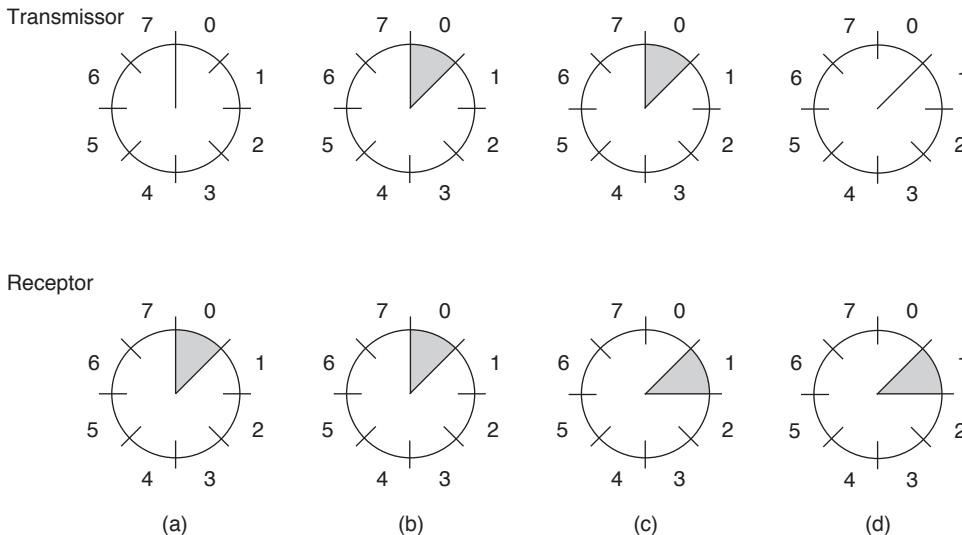



Figura 3.11 | Uma janela deslizante de tamanho 1, com um número de sequência de 3 bits. (a) Inicialmente. (b) Depois que o primeiro quadro é enviado. (c) Depois que o primeiro quadro é recebido. (d) Depois que a primeira confirmação é recebida.

A Figura 3.11 mostra um exemplo com um tamanho máximo de janela igual a 1. Inicialmente, não há quadros pendentes; portanto, os limites inferior e superior da janela do transmissor são iguais, mas, à medida que o tempo passa, a situação se desenvolve da maneira mostrada. Diferentemente da janela do transmissor, a janela do receptor sempre permanece em seu tamanho inicial, incrementando-se à medida que o próximo quadro é aceito e entregue à camada de rede.

3.4.1 Um protocolo de janela deslizante de um bit

Antes de abordarmos o caso geral, vamos examinar primeiro um protocolo de janela deslizante com um tamanho máximo de janela igual a 1. Esse tipo de protocolo utiliza o stop-and-wait, pois o transmissor envia um quadro e aguarda sua confirmação antes de enviar o quadro seguinte.

O Quadro 3.5 representa esse tipo de protocolo. Assim como os demais, ele começa definindo algumas variáveis: *next_frame_to_send* informa qual quadro o transmissor está tentando enviar. De modo semelhante, *frame_expected* informa que quadro o receptor está esperando. Nos dois casos, 0 e 1 são as únicas possibilidades.

Normalmente, uma das duas camadas de enlace de dados do transmissor ou receptor inicia e envia o primeiro quadro. Em outras palavras, apenas um dos programas da camada de enlace de dados pode conter as chamadas das funções *to_physical_layer* e *start_timer* fora do loop principal. A máquina que inicia busca o primeiro pacote em sua camada de rede, constrói um quadro a partir dele e o envia. Quando esse (ou qualquer) quadro chega ao destino, a camada de enlace de dados receptora verifica se ele é uma cópia, como ocorreu no protocolo 3. Se o quadro for o esperado, ele será repassado à camada de rede e a janela do receptor será deslocada para cima.

O campo de confirmação contém o número do último quadro recebido sem erro. Se esse número estiver de acordo com o número de sequência do quadro que o transmissor está tentando enviar, o transmissor saberá que já cuidou do quadro armazenado em *buffer* e poderá buscar o pacote seguinte em sua camada de rede. Se o número de sequência for discordante, o transmissor deverá continuar tentando enviar o mesmo quadro. Sempre que um quadro é recebido, outro quadro também é enviado de volta.

Agora, vamos examinar o protocolo 4 para ver quanto ele é flexível em relação a situações patológicas. Suponha que o computador *A* esteja tentando enviar seu quadro 0 ao computador *B* e que *B* esteja tentando enviar seu quadro 0 ao computador *A*. Imagine que *A* envia um quadro a *B*, mas o intervalo de timeout de *A* é curto demais. Consequentemente, *A* pode completar o timeout repetidas vezes, enviando uma série de quadros idênticos, todos com $seq = 0$ e $ack = 1$.

Quando o primeiro quadro válido chegar a *B*, ele será aceito e *frame_expected* será definido como 1. Todos os quadros subsequentes serão rejeitados, porque *B* agora está esperando quadros com número de sequência 1, e não 0. Além disso, como todas as cópias têm $ack = 1$ e *B* ainda está aguardando uma confirmação de 0, *B* não buscará um novo pacote em sua camada de rede.

Após a chegada de todas as cópias rejeitadas, *B* enviará um quadro para *A* contendo $seq = 0$ e $ack = 0$. Por fim, um desses quadros chegará sem erros à máquina *A*, fazendo com que *A* comece a enviar o próximo pacote. Nenhuma combinação de quadros perdidos ou timeouts prematuros pode fazer o protocolo entregar pacotes duplicados à cama-

```
/* O protocolo 4 (janela deslizante) é bidirecional. */

#define MAX_SEQ 1                          /* deve ser 1 para protocolo 4 */
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"
void protocol4 (void)
{
  seq_nr next_frame_to_send;               /* 0 ou 1 apenas */
  seq_nr frame_expected                    /* 0 ou 1 apenas */
  frame r, s;                              /* variáveis auxiliares */
  packet buffer;                           /* pacote atual sendo enviado */
  event_type event;
  next_frame_to_send = 0;                  /* próximo quadro no fluxo de saída */
  frame_expected = 0;                      /* quadro esperado em seguida */
  from_network_layer(&buffer);             /* busca um pacote da camada de rede */
  s.info = buffer;                         /* prepara para enviar quadro inicial */
  s.seq = next_frame_to_send;              /* insere núm. seq. no quadro */
  s.ack = 1 – frame_expected;              /* confirmação acrescentada */
  to_physical_layer(&s);                   /* transmite o quadro */
  start_timer(s.seq);                      /* inicia a execução do timer */
  while (true) {
    wait_for_event(&event);                /* frame_arrival, cksum_err ou timeout */
    if (event == frame_arrival){           /* um quadro chegou intacto */
      from_physical_layer(&r);             /* vai pegá-lo */
      if (r.seq == frame_expected){        /* trata do fluxo de quadros que chega */
        to_network_layer(&r.info);         /* passa pacote à camada de rede */
        inc(frame_expected);               /* inverte núm. seq. esperado em seguida */
      }
      if (r.ack == next_frame_to_send){    /* trata do fluxo de quadros que sai */
        stop_timer(r.ack);                 /* desliga o timer */
        from_network_layer(&buffer);       /* busca novo pac. da camada de rede */
        inc(next_frame_to_send);           /* inverte núm. seq. do transmissor */
      }
    }
    s.info = buffer;                       /* constrói quadro de saída */
    s.seq = next_frame_to_send;            /* insere núm. seq. nele */
    s.ack = 1 – frame_expected;            /* núm. seq. do último quadro recebido */
    to_physical_layer(&s);                 /* transmite um quadro */
    start_timer(s.seq);                    /* inicia a execução do timer */
  }
}
```

Quadro 3.5 | Um protocolo de janela deslizante de um bit.

da de rede, ignorar um pacote ou chegar a um impasse. O protocolo está correto.

Entretanto, para mostrar quão sutis as interações entre protocolos podem ser, surgirá uma situação peculiar se os dois lados enviarem simultaneamente um pacote inicial. Essa dificuldade de sincronização está ilustrada na Figura 3.12. Na parte (a), é exibida a operação normal do protocolo. Na parte (b), observamos a peculiaridade. Se *B* esperar pelo primeiro quadro de *A* antes de enviar um de seus quadros, a sequência será a da parte (a) e todos os quadros serão aceitos.

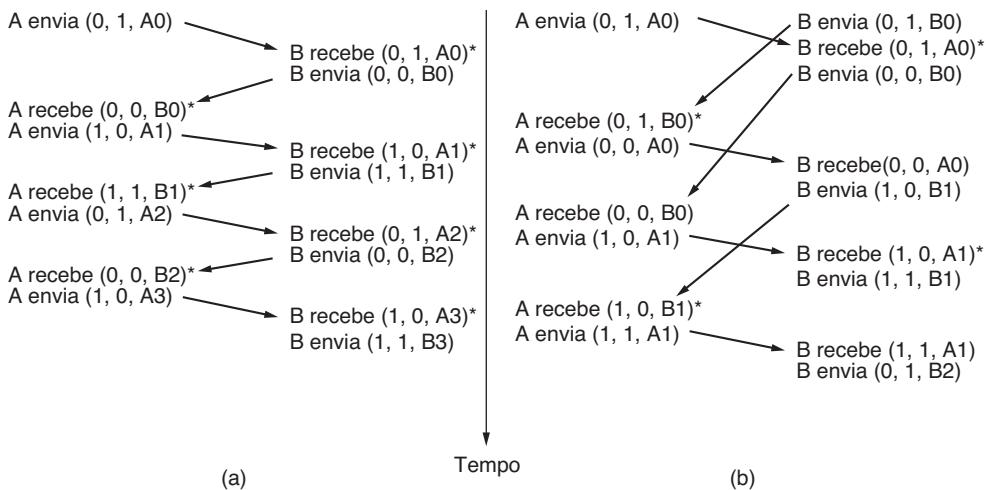



Figura 3.12 | Dois cenários referentes ao protocolo 4. (a) Caso normal. (b) Caso anormal. A notação é sequência, confirmação, número do pacote. Um asterisco indica onde uma camada de rede aceita um pacote.

Porém, se *A* e *B* iniciarem a comunicação ao mesmo tempo, seus primeiros quadros se cruzarão e as camadas de enlace de dados entrarão na situação (b). Em (a), cada quadro recebido traz um novo pacote para a camada de rede; não há cópias. Em (b), metade dos quadros contém cópias, embora não haja erros de transmissão. Situações similares podem ocorrer como resultado de timeouts prematuros, mesmo quando está claro que um lado começa primeiro. Na verdade, se ocorrerem vários timeouts prematuros, os quadros poderão ser enviados três ou mais vezes, desperdiçando uma largura de banda valiosa.

3.4.2 Um protocolo que utiliza go-back-n

Até agora estávamos supondo implicitamente que era insignificante o tempo de transmissão necessário para a chegada de um quadro até o receptor, somado ao tempo de transmissão para o retorno da confirmação. Às vezes, essa suposição é nitidamente falsa. Nessas situações, o longo tempo de viagem de ida e volta pode ter implicações importantes para a eficiência da utilização da largura de banda. Como exemplo, considere um canal de satélite de 50 kbps com um atraso de propagação de ida e volta de 500 ms. Vamos imaginar uma tentativa de usar o protocolo 4 para enviar quadros de 1.000 bits pelo satélite. Em $t = 0$, o transmissor começa a enviar o primeiro quadro. Em $t = 20$ ms, o quadro já foi completamente enviado. Até $t = 270$ ms, o quadro ainda não chegou completamente ao receptor, e até $t = 520$ ms, na melhor das hipóteses, a confirmação não terá voltado ao transmissor (sem nenhum tempo de espera no receptor e com um quadro de confirmação curto). Isso significa que o transmissor esteve bloqueado durante 500/520 ou 96 por cento do tempo (isto é, apenas 4 por cento da largura de banda disponível foi utilizada). É claro que a combinação de um longo tempo de trânsito, alta largura de banda e pequeno comprimento de quadro é desastrosa em termos de eficiência.

O problema descrito anteriormente pode ser visto como uma consequência da regra que exige que um transmissor espere por uma confirmação antes de enviar outro quadro. Se essa restrição não for rigorosa, podemos obter uma eficiência muito melhor. Basicamente, a solução está em permitir que o transmissor envie até *w* quadros antes do bloqueio, e não apenas 1. Com uma escolha apropriada de *w*, o transmissor será capaz de transmitir quadros continuamente, pois as confirmações chegarão aos quadros anteriores antes que a janela se encha, impedindo o bloqueio do transmissor.

Para achar um valor apropriado para *w*, precisamos saber quantos quadros podem caber dentro do canal à medida que se propagam do transmissor ao receptor. Essa capacidade é determinada pela largura de banda em bits/s, multiplicada pelo tempo de trânsito em mão única, ou **produto largura de banda-atraso** do enlace (*BD*). Podemos dividir essa quantidade pelo número de bits em um quadro para expressá-lo como um número de quadros. Chame essa quantidade de *BD*. Então, *w* deve ser definido como $2BD + 1$. O dobro da largura de banda-atraso é o número de quadros que podem estar pendentes se o transmissor enviar quadros continuamente quando o tempo de ida e volta para receber uma confirmação for considerado. O '+1' é porque um quadro de confirmação não será enviado antes que um quadro completo seja recebido.

Para o enlace de exemplo com uma largura de banda de 50 kbps e um tempo de trânsito unidirecional de 250 ms, o produto largura de banda-atraso é de 12,5 kbits ou 12,5 quadros de 1.000 bits cada um. $2BD + 1$ significa, então, 26 quadros. Suponha que o transmissor comece enviando o quadro 0, como antes, e transmita um novo quadro a cada 20 ms. Quando ele tiver terminado de enviar 26 quadros,

em $t = 520$ ms, a confirmação para o quadro 0 terá acabado de chegar. Depois disso, as confirmações chegarão a cada 20 ms, de modo que o transmissor sempre terá permissão para continuar assim que precisar. A partir desse ponto, 25 ou 26 quadros não confirmados sempre estarão pendentes. Em outras palavras, o tamanho de janela máxima do transmissor é 26.

Para tamanhos de janela menores, a utilização do enlace será menor que 100 por cento, pois o transmissor às vezes será bloqueado. Podemos escrever a utilização como a fração de tempo em que o transmissor não está bloqueado:

$$\text{utilização do enlace} \leq \frac{w}{1 + 2BD}$$

Esse valor é um limite superior, pois não leva em consideração nenhum tempo de processamento de quadro e trata o quadro de confirmação como tendo tamanho zero, pois ele normalmente é curto. A equação mostra a necessidade de ter uma janela w grande sempre que o produto de largura de banda-atraso for grande. Se o atraso for alto, o transmissor rapidamente esgotará sua janela, mesmo para uma largura de banda moderada, como no exemplo do satélite. Se a largura de banda for alta, mesmo para um atraso moderado, o transmissor esgotará sua janela rapidamente, a menos que tenha uma janela grande (por exemplo, um enlace de 1 Gbps com atraso de 1 ms mantém 1 megabit). Com o stop-and-wait, para o qual $w = 1$, se houver um atraso de propagação de até mesmo um quadro, a eficiência será menor que 50 por cento.

Essa técnica de manter vários quadros pendentes é um exemplo de **pipelining**. O pipelining de quadros em um canal de comunicação não confiável faz surgir algumas questões muito sérias. Primeiro, o que acontecerá se um quadro em meio a um longo fluxo for danificado ou perdido? Um grande número de quadros sucessivos chegará ao receptor antes mesmo que o transmissor descubra que algo está errado. Quando um quadro danificado chega ao receptor, sem dúvida ele deve ser descartado. No entanto, o que o receptor deve fazer com todos os quadros corretos que o seguem? Lembre-se de que a camada de enlace de dados receptora é obrigada a entregar pacotes à camada de rede em sequência.

Há duas estratégias básicas para lidar com erros na presença do pipelining, ambas mostradas na Figura 3.13.

Em uma opção denominada **go-back-n**, o receptor simplesmente descarta todos os quadros subsequentes e não envia nenhuma confirmação desses quadros descartados. Essa estratégia corresponde a uma janela de recepção de tamanho 1. Em outras palavras, a camada de enlace de dados se recusa a aceitar qualquer quadro, exceto o próximo, que ela tem de entregar à camada de rede. Se a janela do transmissor for totalmente preenchida antes de o timer encerrar a contagem, o pipeline começará a se esvaziar. Consequentemente, o transmissor interromperá a transmissão e retransmitirá todos os quadros não confirmados em ordem, começando pelo quadro danificado ou perdido. Essa abordagem poderá desperdiçar uma grande quantidade de largura de banda se a taxa de erros for alta.

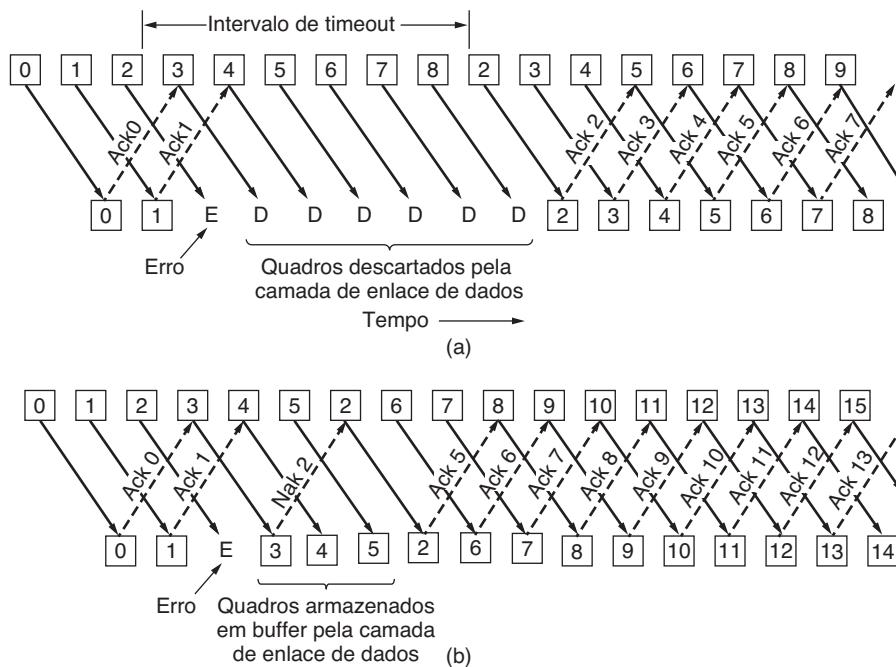



Figura 3.13 | Pipelining e recuperação de erros. Efeito de um erro quando (a) o tamanho da janela receptora é igual a 1 e (b) quando o tamanho da janela receptora é grande.

Na Figura 3.13(a), vemos o go-back-n para o caso em que a janela do receptor é grande. Os quadros 0 e 1 são corretamente recebidos e confirmados. Porém, o quadro 2 está danificado ou perdido. O transmissor, desavisado desse problema, continua a enviar quadros até expirar o timer correspondente ao quadro 2. Em seguida, ele volta até o quadro 2 e começa tudo de novo a partir dele, enviando, mais uma vez, os quadros 2, 3, 4 etc.

A outra estratégia geral para tratamento de erros quando é feito o pipelining de quadros denomina-se **retransmissão seletiva**. Quando ela é utilizada, um quadro incorreto recebido é descartado, mas os quadros sem defeitos recebidos depois dele são aceitos e inseridos no buffer. Quando o transmissor chega ao timeout, apenas o quadro não confirmado mais antigo é retransmitido. Se esse quadro chegar corretamente, o receptor poderá entregar à camada de rede, em sequência, todos os quadros que armazenou no buffer. A estratégia de retransmissão seletiva corresponde a uma janela receptora maior que 1. Caso a janela seja muito grande, essa abordagem poderá exigir um volume de memória muito grande da camada de enlace de dados.

Com frequência, a retransmissão seletiva é combinada com a ação de fazer o receptor enviar uma confirmação negativa, ou NAK (*negative acknowledgement*), ao detectar um erro, por exemplo, quando receber um erro de checksum ou um quadro fora de sequência. As NAKs estimulam a retransmissão antes de expirar o timer correspondente e, desse modo, melhoram o desempenho.

Na Figura 3.13(b), os quadros 0 e 1 são mais uma vez recebidos e confirmados corretamente, e o quadro 2 é perdido. Quando o quadro 3 chega ao receptor, a camada de enlace de dados do receptor percebe que perdeu um quadro e, assim, envia de volta uma NAK correspondente ao quadro 2, mas armazena o quadro 3 no buffer. Quando os quadros 4 e 5 chegam, eles também são inseridos no buffer pela camada de enlace de dados, em vez de ser repassados à camada de rede. Por fim, a NAK do quadro 2 volta ao transmissor, que retransmite de imediato o quadro 2. Quando esse quadro chega, a camada de enlace de dados fica com os quadros 2, 3, 4 e 5, e pode repassar todos eles à camada de rede na ordem correta. Ela também pode confirmar todos os quadros, inclusive até o 5, como mostra a figura. Se a NAK se perder, o transmissor chegará ao timeout correspondente ao quadro 2 e o enviará (e apenas esse quadro) por sua própria iniciativa, mas isso pode acontecer um pouco mais tarde.

Esses dois enfoques alternativos são dilemas entre uso eficiente de largura de banda e espaço no buffer da camada de enlace de dados. Dependendo de qual recurso seja mais escasso, um ou outro poderá ser usado. O Quadro 3.6 mostra um protocolo go-back-n no qual a camada de enlace de dados receptora aceita apenas quadros em ordem; os quadros que vierem depois de um quadro com erro serão descartados. Nesse protocolo, pela primeira vez abandonamos a suposição de que a camada de rede sempre tem um suprimento infinito de pacotes a enviar. Quando a camada de rede tem um pacote que deseja enviar, ela pode provocar a ocorrência de um evento *network_layer_ready*. Entretanto, para reforçar o limite de controle de fluxo sobre a janela do transmissor ou o número de quadros não confirmados pendentes em qualquer instante, a camada de enlace de dados deve ser capaz de proibir a camada de rede de sobrecarregá-la com mais trabalho. As funções de biblioteca *enable_network_layer* e *disable_network_layer* executam essa funcionalidade.

Observe que o número máximo de quadros que podem estar pendentes em qualquer instante não é o mesmo que o tamanho do espaço do número de sequência. Para go-back-n, *MAX_SEQ* quadros podem estar pendentes em qualquer instante, mesmo que haja *MAX_SEQ* + 1 números de sequência distintos (que são 0, 1, 2,..., *MAX_SEQ*). Veremos uma restrição ainda maior para o protocolo seguinte, a retransmissão seletiva. Para saber por que essa restrição é necessária, considere a situação a seguir, com *MAX_SEQ* = 7:

1. O transmissor envia quadros de 0 a 7.
2. Uma confirmação com piggyback (de carona) para o quadro 7 volta ao transmissor.
3. O transmissor envia mais oito quadros, novamente com números de sequência de 0 a 7.
4. Agora chega outra confirmação com piggyback correspondente ao quadro 7.

A questão é: os oito quadros pertencentes ao segundo lote chegaram com sucesso, ou todos eles se perderam (a contagem descarta os quadros posteriores a um erro, considerando-os perdidos)? Nos dois casos, o receptor enviaria o quadro 7 como confirmação. O transmissor não tem como saber disso. Por essa razão, o número máximo de quadros pendentes deve se restringir a *MAX_SEQ*.

Apesar de não armazenar no buffer os quadros recebidos após um quadro com erro, o protocolo 5 não escapa totalmente ao problema do armazenamento em buffer. Tendo em vista que um transmissor talvez seja obrigado a retransmitir todos os quadros não confirmados em determinado momento no futuro, ele deverá reter todos os quadros transmitidos até ter certeza de que eles foram aceitos pelo receptor. Quando uma confirmação chega para o quadro n, os quadros $n - 1$, $n - 2$ e assim por diante também são confirmados de forma automática. Esse tipo de confirmação é chamado de **confirmação cumulativa**. Essa propriedade é especialmente importante nos casos em que alguns dos quadros anteriores que representavam confirmações se perderam ou foram adulterados. Sempre que chega uma confirmação, a camada de enlace de dados verifica se algum buffer pode ser liberado. Se os buffers puderem ser liberados (isto é, se houver espaço disponível

```
/* O protocolo 5 (go-back-n) permite a existência de muitos quadros pendentes. O transmissor poderá transmitir até
   MAX_SEQ quadros sem a necessidade de esperar por uma confirmação. Além disso, ao contrário dos protocolos
   anteriores, não presumimos que a camada de rede esteja sempre recebendo um novo pacote. Em vez disso, a camada
   de rede provoca o evento network_layer_ready quando há um pacote a ser enviado. */

#define MAX_SEQ 7
typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready}event_type;
#include "protocol.h"

static boolean between(seq_nr a, seq_nr b, seq_nr c)
{
/* Retorna true se a <= b < c de modo circular; caso contrário, false. */
  if (((a <= b) && (b < c)) || ((c < a) && (a <= b)) || ((b < c) && (c < a)))
    return(true);
  else
    return(false);
}

static void send_data(seq_nr frame_nr,seq_nr frame_expected,packet buffer[ ])
{
/* Constrói e envia um quadro de dados. */
  frame s;                                              /* variável auxiliar */

  s.info = buffer[frame_nr];                            /* insere pacote no quadro */
  s.seq = frame_nr;                                     /* insere núm. seq. no quadro */
  s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1);   /* confirm. piggyback */
  to_physical_layer(&s);                                /* transmite o quadro */
  start_timer(frame_nr);                                /* inicia o timer */
}

void protocol5(void)
{
  seq_nr next_frame_to_send;                            /* MAX_SEQ> 1; usado para fluxo de saída */
  seq_nr ack_expected;                                  /* quadro mais antigo como ainda não confirmado */
  seq_nr frame_expected;                                /* próximo quadro esperado no fluxo de entrada */
  frame r;                                              /* variável auxiliar */
  packet buffer[MAX_SEQ + 1];                           /* buffers para o fluxo de saída */
  seq_nr nbuffered;                                     /* número de buffers de saída atualmente em uso */
  seq_nr i;                                             /* usado para indexar no array de buffers */
  event_type event;

  enable_network_layer();                               /* permite network_layer_ready events */
  ack_expected = 0;                                     /* próximo ack esperado na entrada */
  next_frame_to_send = 0;                               /* próximo quadro saindo */
  frame_expected= 0;                                    /* número do quadro esperado na entrada */
  nbuffered = 0;                                        /* inicialmente, nenhum pacote no buffer */

  while (true) {
    wait_for_event(&event);                             /* quatro possibilidades: ver event_type acima */
    switch(event) {
      case network_layer_ready:                         /* a camada de rede tem um pacote a enviar */
        /* Aceita, salva e transmite um novo quadro. */
        from_network_layer(&buffer[next_frame_to_send]);   /* busca novo pacote */
```

Quadro 3.6 | Um protocolo de janela deslizante que utiliza go-back-n. (*continua*)

(*continuação*)

```
        nbuffered = nbuffered + 1;                          /* expande a janela do transmissor */
        send_data(next_frame_to_send, frame_expected, buffer); /* transmite o quadro */
        inc(next_frame_to_send);                            /* avança o limite da janela superior do transmissor */
        break;

      case frame_arrival:                                   /* um quadro de dados ou controle chegou */
        from_physical_layer(&r);                            /* pega quadro que chegou da camada física */

        if (r.seq == frame_expected){
          /* Quadros são aceitos apenas em ordem */
          to_network_layer(&r.info);                        /* passa pacote à camada de rede */
          inc(frame_expected);                              /* avança o limite inferior da janela do receptor */
        }

        /* Ack n implica n – 1, n – 2 etc. Verifica isso. */
        while (between(ack_expected, r.ack, next_frame_to_send)){
          /* Trata o ack piggyback. */
          nbuffered = nbuffered – 1;                        /* um quadro a menos em buffer */
          stop_timer(ack_expected);                         /* quadro chegou intacto; interrompe timer */
          inc(ack_expected);                                /* ajusta a janela do transmissor */
        }
        break;

        case cksum_err: break;                              /* apenas ignora quadros ruins */

        case timeout:                                       /* problema; retransmite todos os quadros pendentes */
          next_frame_to_send = ack_expected;                /* inicia retransmissão aqui */
          for (i = 1; i <= nbuffered; i++) {
            send_data(next_frame_to_send, frame_expected, buffer);/* reenvia quadro */
            inc(next_frame_to_send);                        /* prepara para enviar o quadro seguinte */
            }

        }

      if (nbuffered < MAX_SEQ)
            enable_network_layer();
      else
            disable_network_layer();
  }
}
```

Quadro 3.6 | Um protocolo de janela deslizante que utiliza go-back-n.

na janela), uma camada de rede bloqueada anteriormente poderá ter permissão para provocar mais eventos *network_layer_ready*.

Para esse protocolo, supomos que sempre existe tráfego no sentido inverso, para que as confirmações possam ser transportadas por piggyback. O protocolo 4 não precisa dessa suposição, pois ele envia um quadro de volta toda vez que recebe um quadro, mesmo que já tenha enviado esse quadro. No próximo protocolo, resolveremos de modo elegante o problema do tráfego de mão única.

Por ter vários quadros pendentes, é claro que o protocolo 5 necessita de vários timers, um para cada quadro pendente. Cada quadro tem um timeout independente de todos os demais. Porém, todos esses timers podem ser facilmente simulados por software, usando-se um único timer do hardware para provocar interrupções periódicas. Os timeouts pendentes formam uma lista encadeada, com cada nó da lista contendo a quantidade de pulsos do timer até que ele expire, o quadro sendo sincronizado e um ponteiro para o nó seguinte.

Para ilustrar como os timers poderiam ser implementados, considere o exemplo da Figura 3.14(a). Suponha que o timer pulse uma vez a cada 1 ms. Inicialmente, o tempo real é 10:00:00.000 e há três timeouts pendentes, em 10:00:00.005, 10:00:00.013 e 10:00:00.019. Toda vez que o timer do hardware pulsar, o tempo real será atualizado e o contador de pulsos no início da lista será decrementado. Quando o contador de pulsos for igual a zero, ocorrerá um timeout e o nó será removido da lista, como mostra a Figura 3.14(b). Embora essa organização exija que a lista seja examinada quando *start_timer* ou *stop_timer* são chamadas, ela não requer muito trabalho por pulso. No protocolo 5, essas duas funções receberam um parâmetro, que indica o quadro a ser sincronizado.

3.4.3 Um protocolo que utiliza retransmissão seletiva

O protocolo go-back-n funciona bem quando há poucos erros, mas, se a linha estiver muito ruidosa, ele desperdiçará muita largura de banda com os quadros retransmitidos. Uma estratégia alternativa, o protocolo de retransmissão seletiva, é permitir que o receptor aceite e coloque no buffer os quadros subsequentes a um danificado ou perdidos.

Nesse protocolo, tanto o transmissor quanto o receptor mantêm uma janela de números de sequência pendentes e aceitáveis, respectivamente. O tamanho da janela do transmissor é medido a partir de 0 e atinge um número máximo predefinido. Por outro lado, a janela do receptor tem sempre um tamanho fixo e igual ao máximo predeterminado. O receptor tem um buffer reservado para cada número de sequência dentro de sua janela fixa. Associado a cada buffer há um bit (*arrived*) que informa se o buffer está cheio ou vazio. Sempre que um quadro chega, seu número de sequência é verificado pela função *between*, para confirmar se ele se enquadra na janela. Se isso ocorrer e o quadro ainda não tiver sido recebido, ele será aceito e armazenado. Essa ação é executada sem levar em conta se o quadro contém ou não o próximo pacote esperado pela camada de rede. É obvio que ele deve ser mantido dentro da camada de enlace de dados e não deve ser repassado à camada de rede até que todos os quadros de números mais baixos já tenham sido entregues a essa camada na ordem correta. Um protocolo que utiliza esse algoritmo é apresentado no Quadro 3.7.

A recepção não sequencial introduz determinados problemas que não estão presentes em protocolos nos quais os quadros só são aceitos em ordem. Podemos ilustrar melhor o problema com um exemplo. Imagine que haja um número de sequência de 3 bits, de modo que o transmissor tenha permissão para transmitir até sete quadros antes de ser obrigado a esperar por uma confirmação. Inicialmente, as janelas do transmissor e do receptor são semelhantes às da Figura 3.15(a). No momento, o transmissor envia os quadros de 0 a 6. A janela do receptor permite que ele aceite qualquer quadro com número de sequência entre 0 e 6, ambos inclusive. Todos os sete quadros chegam corretamente; assim, o receptor os confirma e avança a janela para permitir a recepção de 7, 0, 1, 2, 3, 4 ou 5, conforme mostra a Figura 3.15(b). Todos os sete buffers são marcados como vazios.

Nesse ponto ocorre o desastre, na forma de um raio que atinge a central telefônica e apaga todas as confirmações. O protocolo deve operar corretamente apesar desse desastre. Mais tarde, o transmissor entra em timeout e retransmite o quadro 0. Quando esse quadro chega ao receptor, é feita uma conferência para ver se ele se ajusta à janela do receptor. Infelizmente, na Figura 3.15(b), o quadro 0 está dentro da nova janela e, assim, ele será aceito como um novo quadro. O receptor também envia uma confirmação com piggyback para o quadro 6, pois os quadros de 0 a 6 foram recebidos.

O transmissor fica feliz em saber que todos os quadros transmitidos realmente chegaram de forma correta; portanto, ele avança sua janela e envia imediatamente os quadros 7, 0, 1, 2, 3, 4 e 5. O quadro 7 será aceito pelo receptor e seu pacote será repassado diretamente à camada de rede. Logo depois, a camada de enlace de dados receptora verifica se já tem um quadro 0 válido, descobre que sim e repassa o pacote incorporado à camada de rede como se fosse um novo. Consequentemente, a camada de rede recebe um pacote incorreto e o protocolo falha.

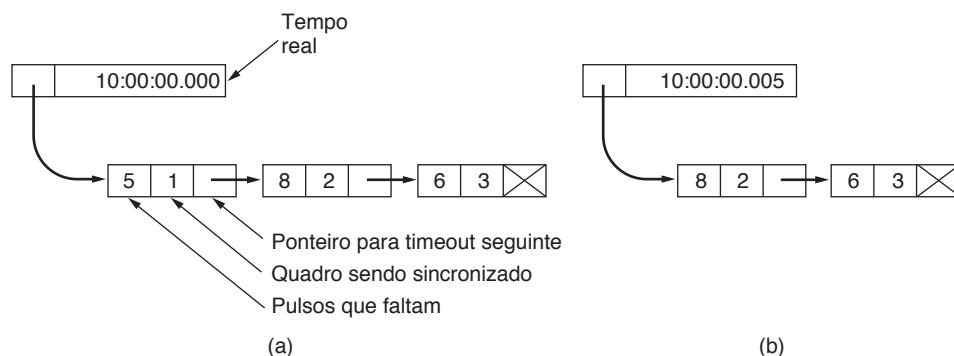



Figura 3.14 | Simulação de vários timers por software. (a) Os timeouts enfileirados. (b) A situação após o primeiro timeout expirar.

```
/* O protocolo 6 (retransmissão seletiva) aceita quadros fora de ordem, mas repassa pacotes para a camada de rede
   obedecendo à ordem de transmissão. Há um timer associado a cada quadro pendente. Quando o timer expira, apenas o
   quadro que o contém é retransmitido, e não todos os quadros pendentes, como ocorria no protocolo 5. */

#define MAX_SEQ 7                                         /* deve ser 2ˆn – 1 */
#define NR_BUFS ((MAX_SEQ + 1)/2)
typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready, ack_timeout} event_type;
#include "protocol.h"
boolean no_nak = true;                                    /* nenhuma nak foi enviada */
seq_nr oldest_frame = MAX_SEQ + 1;                        /* valor inicial é apenas para o simulador */

static boolean between(seq_nr a, seq_nr b, seq_nr c)
{
/* O mesmo que no protocolo 5, porém mais curto e mais obscuro. */
  return ((a <= b) && (b < c)) || ((c < a) && (a <= b)) || ((b < c) && (c < a));
}

static void send_frame(frame_kind fk, seq_nr frame_nr, seq_nr frame_expected, packet buffer[ ])
{
/* Constrói e envia um quadro data, ack ou nak. */
  frame s;                                                /* variável auxiliar */

  s.kind = fk;                                            /* kind == data, ack, ou nak */
  if (fk == data) s.info = buffer[frame_nr % NR_BUFS];
  s.seq = frame_nr;                                       /* significativo apenas para quadros de dados */
  s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1);
  if (fk == nak) no_nak = false;                          /* um nak por quadro, por favor */
  to_physical_layer(&s);                                  /* transmite o quadro */
  if (fk == data) start_timer(frame_nr % NR_BUFS);
  stop_ack_timer();                                       /* não necessário para quadro ack separado */
}
void protocol6(void)
{
  seq_nr ack_expected;                                    /* limite inferior da janela do transmissor */
  seq_nr next_frame_to_send;                              /* limite superior da janela do transmissor + 1 */
  seq_nr frame_expected;                                  /* limite inferior da janela do receptor */
  seq_nr too_far;                                         /* limite superior da janela do receptor + 1 */
  int i;                                                  /* índice para os buffers de entrada/saída */
  frame r;                                                /* variável auxiliar */
  packet out_buf[NR_BUFS];                                /* buffers para o fluxo de saída */
  packet in_buf[NR_BUFS];                                 /* buffers para o fluxo de entrada */
  boolean arrived[NR_BUFS];                               /* mapa de bits de entrada */
  seq_nr nbuffered;                                       /* quantidade de buffers de saída usados atualmente */
  event_type event;

  enable_network_layer();                                 /* inicializa */
  ack_expected = 0;                                       /* próximo ack esperado no fluxo de entrada */
  next_frame_to_send = 0;                                 /* número do próximo quadro de saída */
  frame_expected = 0;
  too_far = NR_BUFS;
  nbuffered = 0;                                          /* inicialmente nenhum pacote em buffer */
  for (i = 0; i < NR_BUFS; i++) arrived[i] = false;
```

Quadro 3.7 | Um protocolo de janela deslizante que utiliza a retransmissão seletiva. (*continua*)

(*continuação*)

```
while (true) {
  wait_for_event(&event);                                    /* cinco possibilidades: ver event_type acima */
  switch(event) {
    case network_layer_ready:                                /* aceita, salva e transmite um novo quadro */
      nbuffered = nbuffered + 1;                             /* estende a janela */
      from_network_layer(&out_buf[next_frame_to_send % NR_BUFS]); /* busca novo pacote */
      send_frame(data, next_frame_to_send, frame_expected, out_buf); /* transmite o quadro */
      inc(next_frame_to_send);                               /* avança o limite da janela superior */
      break;
    case frame_arrival:                                      /* um quadro de dados ou de controle chegou */
      from_physical_layer(&r);                               /* busca quadro que chega da camada física */
      if (r.kind == data) {
        /* Um quadro correto chegou. */
        if ((r.seq != frame_expected) && no_nak)
          send_frame(nak, 0, frame_expected, out_buf); else start_ack_timer();
        if (between(frame_expected,r.seq,too_far) && (arrived[r.seq%NR_BUFS]==false)) {
          /* quadros podem ser aceitos em qualquer ordem. */
          arrived[r.seq % NR_BUFS] = true;                   /* marca o buffer como cheio */
          in_buf[r.seq % NR_BUFS] = r.info;                  /* insere dados no buffer */
          while (arrived[frame_expected % NR_BUFS]) {
            /* Passa quadros e avança a janela. */
            to_network_layer(&in_buf[frame_expected % NR_BUFS]);
            no_nak = true;
            arrived[frame_expected % NR_BUFS] = false;
            inc(frame_expected);                             /* avança o limite inferior da janela do receptor */
            inc(too_far);                                    /* avança o limite superior da janela do receptor */
            start_ack_timer();                               /* para ver se precisa de ack separado */
          }
        }
      }
      if((r.kind==nak) && between(ack_expected,(r.ack+1)%(MAX_SEQ+1),next_frame_to_send))
        send_frame(data, (r.ack+1) % (MAX_SEQ + 1), frame_expected, out_buf);

      while (between(ack_expected, r.ack, next_frame_to_send)) {
        nbuffered = nbuffered – 1;                           /* trata do ack piggyback */
        stop_timer(ack_expected % NR_BUFS);                  /* quadro chegou intacto */
        inc(ack_expected);                                   /* avança o limite inferior da janela do transmissor */
      }
      break;
  case cksum_err:
      if (no_nak) send_frame(nak, 0, frame_expected, out_buf);   /* quadro danificado */
      break;
  case timeout:
      send_frame(data, oldest_frame, frame_expected, out_buf);   /* timeout */
      break;
  case ack_timeout:
      send_frame(ack,0,frame_expected, out_buf);             /* timer de ack expirou; envia ack */
  }
  if (nbuffered < NR_BUFS) enable_network_layer(); else disable_network_layer();
 }
}
```

Quadro 3.7 | Um protocolo de janela deslizante que utiliza a retransmissão seletiva.

A essência do problema é que, depois que o receptor avançou a janela, a nova faixa de números de sequência válidos substituiu a antiga. Consequentemente, o próximo lote de quadros poderia ser formado por cópias (se todas as confirmações se perderam) ou por novos quadros (se todas as confirmações foram recebidas). O receptor não tem como distinguir esses dois casos.

A saída desse dilema reside em ter certeza de que, depois que o receptor avançou sua janela, não há sobreposição entre essa janela e a original. Para assegurar que não há sobreposição, o tamanho máximo da janela deve ser igual à metade do intervalo dos números de sequência, como ocorre na Figura 3.15(c) e (d). Com 3 bits, os números de sequência vão variar de 0 a 7. Apenas quatro quadros não confirmados devem estar pendentes em qualquer instante. Dessa forma, se o receptor só tiver aceito os quadros de 0 a 3 e avançado sua janela para aceitar os quadros de 4 a 7, ele poderá saber, sem nenhuma dúvida, se os quadros subsequentes são retransmissões (0 a 3) ou novos quadros (4 a 7). Em geral, o tamanho da janela para o protocolo 6 será $(MAX_SEQ + 1)/2$.

Uma pergunta interessante é: quantos buffers o receptor deverá ter? De maneira alguma ele aceitará quadros cujos números de sequência estejam abaixo do limite inferior da janela ou acima do limite superior. Consequentemente, o número de buffers necessário é igual ao tamanho da janela, e não ao intervalo dos números de sequência. No exemplo anterior de um número de sequência de 3 bits, são necessários quatro buffers, numerados de 0 a 3. Quando o quadro i chega, ele é colocado no buffer i mod 4. Observe que, apesar de i e $(i + 4)$ mod 4 estarem 'competindo' pelo mesmo buffer, eles nunca estão dentro da janela ao mesmo tempo, pois isso implicaria um tamanho de janela de no mínimo 5.

Pela mesma razão, o número de timers necessários é igual ao número de buffers, e não ao tamanho do espaço de sequência. Efetivamente, um timer é associado a cada buffer. Quando o timer chega a seu timeout, o conteúdo do buffer é retransmitido.

O protocolo 6 também alivia a suposição implícita de que o canal está muito carregado. Fizemos essa suposição no protocolo 5, quando contamos com quadros enviados na direção inversa das confirmações com piggyback. Se o tráfego inverso for leve, a confirmação será retida por um longo período, o que poderá causar problemas. No limite, se houver um tráfego intenso em um sentido e nenhum tráfego no outro, então o protocolo será bloqueado quando a janela transmissora atingir seu máximo.

Para aliviar essa suposição, após um quadro de dados sequencial ser recebido, um timer auxiliar é iniciado por *start_ack_timer*. Se nenhum tráfego inverso tiver se apresentado antes que esse timer expire, será enviado um quadro de confirmação separado. Uma interrupção provocada pelo timer auxiliar é chamada evento *ack_timeout*. Diante dessa organização, o fluxo de tráfego unidirecional passa a ser possível nesse momento, pois a falta de quadros de dados inversos nos quais as confirmações possam ser transportadas não representa mais um obstáculo. Existe apenas um timer auxiliar e, se *start_ack_timer* for chamada durante o intervalo em que o timer estiver funcionando, ela não terá efeito. O timer não é reiniciado ou estendido, pois sua finalidade é oferecer uma taxa mínima de confirmações.

É essencial que o timeout associado ao timer auxiliar seja ligeiramente mais curto que o timer utilizado para sincronizar quadros de dados. Essa condição é necessária para assegurar que a confirmação de um quadro corretamente recebido chegue antes de o timer de retransmissão do quadro expirar, de modo que o transmissor não tenha de retransmiti-lo.

O protocolo 6 utiliza uma estratégia mais eficiente que o protocolo 5 para tratamento de erros. Sempre que tem motivos para suspeitar da ocorrência de um erro, o receptor envia um quadro de confirmação negativa (NAK) de volta ao transmissor. Esse quadro é um pedido de retransmissão do quadro especificado na NAK. Existem dois casos que podem provocar a suspeita do receptor: a chegada de um quadro danificado ou de um quadro diferente do esperado (quadro potencialmente perdido). Para impedir que sejam feitas várias solicitações de retransmissão do mesmo quadro perdido, o receptor deve controlar se já foi enviada uma NAK correspondente a determinado quadro. A variável *no_nak* no protocolo 6 será verdadeira (*true*) se nenhuma NAK tiver sido enviada para *frame_expected*. Se a NAK for danificada ou per-

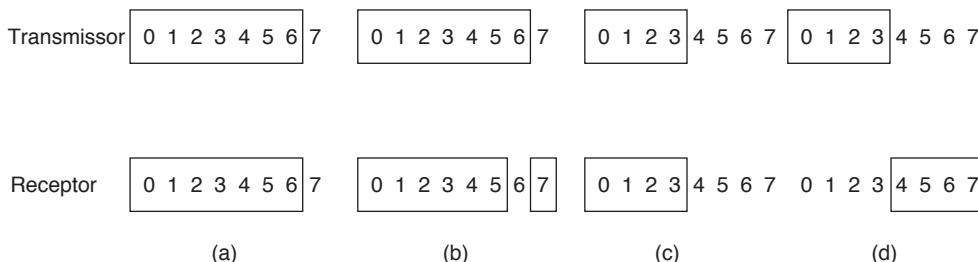



Figura 3.15 | (a) Situação inicial com uma janela de tamanho 7. (b) Após sete quadros serem transmitidos e recebidos, mas não confirmados. (c) Situação inicial com uma janela de tamanho 4. (d) Após quatro quadros serem transmitidos e recebidos, mas não confirmados.

dida, não haverá nenhum prejuízo real, pois, com o término do intervalo de timeout, de qualquer forma, o transmissor retransmitirá o quadro ausente. Se o quadro errado chegar depois que uma NAK tiver sido enviada e perdida, *no_nak* será verdadeira e o timer auxiliar será inicializado. Quando o timer expirar, uma ACK será enviada para ressincronizar o transmissor com o status atual do receptor.

Em algumas situações, o tempo necessário para que um quadro se propague até o destino, seja processado lá e tenha a confirmação retornada é (praticamente) constante. Nessas situações, o transmissor pode ajustar seu timer para ser 'estreito', apenas ligeiramente maior que o intervalo normal esperado entre o envio de um quadro e a recepção de sua confirmação. As NAKs não são úteis nesse caso.

Porém, em outras situações o tempo deve ser bastante variável. Por exemplo, se o tráfego inverso for esporádico, o tempo antes da confirmação será mais curto quando houver tráfego inverso, e mais longo quando não houver. O transmissor enfrenta a escolha de definir o intervalo para um valor pequeno (arriscando retransmissões desnecessárias) ou defini-lo para um valor grande (e ficar ocioso por longo período após um erro). As duas opções desperdiçam largura de banda. Em geral, sempre que o desvio-padrão do intervalo de confirmação é grande em comparação com o próprio intervalo, o timer pode ser ajustado 'mais livremente' para ser conservador. As NAKs podem, então, acelerar bastante a retransmissão de quadros perdidos ou danificados.

Um problema intimamente relacionado com o uso de timeouts e NAKs é a questão de determinar o quadro que provocou um timeout. No protocolo 5, ele é sempre *ack_expected*, porque é sempre o mais antigo. No protocolo 6, não há nenhuma forma trivial para determinar o quadro que chegou em timeout. Imagine que os quadros de 0 a 4 tenham sido transmitidos, significando que a lista de quadros pendentes é 01234, na ordem do mais antigo para o mais recente. Agora, imagine que o quadro 0 chegue em timeout, que 5 (um novo quadro) seja transmitido, 1 e 2 cheguem em timeout e 6 (outro quadro novo) seja transmitido. Nesse ponto, a lista de quadros pendentes será 3405126, do mais antigo para o mais recente. Se todo o tráfego de chegada (isto é, quadros que transportam confirmações) for perdido durante algum tempo, esses sete quadros pendentes chegarão em timeout nessa ordem.

Para evitar que o exemplo fique ainda mais complicado do que já está, não mostramos a administração do timer. Em vez disso, consideramos apenas que a variável *oldest_frame* é ativada no momento de timeout para indicar o quadro que chegou em timeout.

3.5 Exemplos de protocolos de enlace de dados

Dentro de um prédio, as LANs são muito utilizadas para interconexão, mas a maior parte da infraestrutura de rede a longa distância é montada a partir de linhas ponto a ponto. No Capítulo 4, examinaremos melhor as LANs. Aqui, em duas situações comuns, veremos os protocolos de enlace de dados nas linhas ponto a ponto da Internet. A primeira situação é quando os pacotes são enviados por enlaces de fibra óptica SONET, nas redes a longas distâncias. Esses enlaces são muito utilizados, por exemplo, para conectar roteadores nos diferentes locais da rede de um ISP.

A segunda situação é a dos enlaces ADSL que se localizam no circuito terminal da rede telefônica, na borda da Internet. Esses enlaces conectam milhões de indivíduos e empresas à Internet.

A Internet precisa de enlaces ponto a ponto para esses fins, bem como modems de linha discada, linhas alugadas, modems a cabo e assim por diante. Um protocolo-padrão, chamado **protocolo ponto a ponto**, ou **PPP** (**Point-to-Point Protocol**), é usado para enviar pacotes por esses enlaces. O PPP é definido na RFC 1661 e mais elaborado na RFC 1662 e em outras RFCs (Simpson, 1994a, 1994b). Enlaces SONET e ADSL aplicam PPP, mas de maneiras diferentes.

3.5.1 Pacotes sobre SONET

O SONET, que analisamos na Seção 2.6.4, é o protocolo da camada física mais utilizado pelos enlaces de fibra óptica que compõem o backbone das redes de comunicação, incluindo o sistema telefônico. Ele oferece um fluxo de bits que trabalha em uma taxa bem definida, por exemplo, 2,4 Gbps para um enlace OC-48. Esse fluxo de bits é organizado como cargas úteis de bytes, com tamanho fixo, que se repetem a cada 125 μs, havendo ou não dados do usuário para transmitir.

Para transportar pacotes por esses enlaces, é preciso que haja algum mecanismo de enquadramento para distinguir pacotes ocasionais do fluxo de bits contínuo no qual são transportados. O PPP atua sobre roteadores IP para fornecer esse mecanismo, como mostra a Figura 3.16.

O PPP é uma melhoria de um protocolo mais antigo e mais simples, chamado **SLIP** (**Serial Line Internet Protocol**), e é usado para lidar com a configuração do enlace de detecção de erro, dar suporte a múltiplos protocolos, permitir autenticação e outros. Com um grande conjunto de opções, o PPP dispõe de três recursos principais:

1. Um método de enquadramento que delineia de forma não ambígua o fim de um quadro e o início do seguinte. O formato do quadro também lida com a detecção de erros.
2. Um protocolo de controle de enlace usado para ativar linhas, testá-las, negociar opções e desativá-las novamente de forma controlada quando não forem mais necessárias. Esse protocolo é denominado protocolo de controle de enlace, ou **LCP** (**Link Control Protocol**).

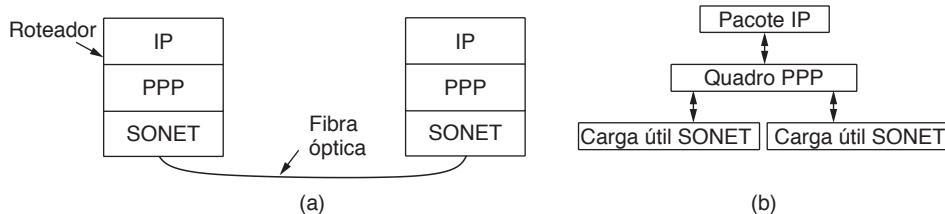



Figura 3.16 | Pacotes sobre SONET. (a) Uma pilha de protocolos. (b) Relacionamentos de quadros.

3. Um modo de negociar as opções da camada de rede de modo independente do protocolo da camada a ser utilizado. O método escolhido deve ter um protocolo de controle de rede, ou **NCP** (**Network Control Protocol**), diferente para cada camada de rede aceita.

O formato de quadro PPP foi definido de modo a ter uma aparência semelhante ao formato de quadro **HDLC** (**High-level Data Link Control**), uma instância bastante utilizada de uma família de protocolos mais antiga, pois não há motivo algum para a definição de um novo padrão.

A principal diferença entre o PPP e o HDLC é que o primeiro é orientado a caracteres, e não a bits. Especificamente, o PPP utiliza a técnica de inserção de bytes e todos os quadros representam um número inteiro de bytes. Já o HDLC usa a inserção de bits, e permite quadros de, digamos, 30,25 bytes.

Porém, na prática, há uma segunda diferença importante. O HDLC oferece transmissão confiável com uma janela deslizante, confirmações e timeouts da forma como estudamos. O PPP também pode oferecer transmissão confiável em ambientes com ruído, como redes sem fio; os detalhes exatos são definidos na RFC 1663. Entretanto, isso raramente é feito na prática. Em vez disso, um padrão adotado 'ainda sem número' quase sempre é usado na Internet para oferecer serviço não orientado a conexão sem confirmação.

A Figura 3.17 mostra o formato do quadro PPP. Todos os quadros PPP começam com o byte de flag padrão do HDLC, 0x7E (01111110). O byte de flag é inserido se ocorrer dentro do campo de *Carga útil* usando o byte de escape 0x7D. O byte seguinte é o XOR entre o byte de escape e 0x20, que inverte o 5º bit. Por exemplo, 0x7D 0x5E é a sequência de escape para o byte de flag 0x7E. Isso significa que o início e o final dos quadros podem ser pesquisados simplesmente procurando-se pelo byte 0x7E, pois ele não ocorrerá em nenhum outro lugar. A regra de retirada na recepção de um quadro é procurar por 0x7D, removê-lo e realizar o XOR do byte seguinte com 0x20. Além disso, apenas um byte de flag é necessário entre os quadros. Vários bytes de flag podem ser usados para preencher o enlace quando não existem quadros para transmitir.

Após o byte de flag de início do quadro, temos o campo *Endereço*, que sempre é definido como o valor binário 11111111, indicando que todas as estações devem aceitar o quadro. A utilização desse valor evita o problema da necessidade de atribuição de endereços de enlace de dados.

O campo *Controle* vem depois do campo *Endereço* e seu valor-padrão é 00000011. Esse valor indica um quadro não numerado. Como os campos *Endereço* e *Controle* sempre são constantes na configuração-padrão, o LCP oferece o mecanismo necessário para as duas partes negociarem uma opção que os omita completamente e economize 2 bytes por quadro.

O quarto campo do quadro PPP é o *Protocolo*. Sua tarefa é informar o tipo de pacote que se encontra no campo *Carga útil*. Os códigos que começam com o bit 0 são definidos para o IP versão 4, IP versão 6 e outros protocolos da camada de rede que poderiam ser usados, como IPX e AppleTalk. Os códigos que começam com o bit 1 são utilizados para protocolos de configuração do PPP, incluindo LCP e um NCP diferente para cada protocolo da camada de rede admitido. O tamanho-padrão do campo *Protocolo* é 2 bytes, mas é possível negociar uma redução para 1 byte, utilizando-se o LCP. Os projetistas talvez tenham sido demasiadamente cuidadosos, achando que algum dia poderia haver mais de 256 protocolos em uso.

O campo *Carga útil* tem comprimento variável, podendo se estender até o tamanho máximo negociado. Se o comprimento não for negociado com o uso do LCP durante a configuração da linha, será empregado um comprimento-padrão

Figura 3.17 | O formato completo do quadro PPP para a operação no modo não numerado.

de 1.500 bytes. Poderá haver um preenchimento logo após a carga útil, caso seja necessário.

Depois do campo *Carga útil*, temos o campo *Checksum*, que normalmente tem 2 bytes, embora seja possível negociar um checksum de 4 bytes, o qual, na verdade, é o mesmo CRC de 32 bits cujo polinômio gerador é mostrado no final da Seção 3.2.2. O checksum de 2 bytes também é um CRC-padrão.

O PPP é um mecanismo de enquadramento que pode transportar os pacotes de vários protocolos por muitos tipos de camadas físicas. Para usar PPP sobre SONET, as escolhas a fazer estão descritas na RFC 2615 (Malis e Simpson, 1999). É utilizado um checksum de 4 bytes, pois esse é o meio principal de detectar erros de transmissão pelas camadas física, de enlace e de rede. Recomenda-se que os campos *Endereço*, *Controle* e *Protocolo* não sejam compactados, pois os enlaces SONET já trabalham com velocidades relativamente altas.

Também há um recurso incomum. A carga útil PPP é misturada (conforme descrevemos na Seção 2.5.1) antes de ser inserida na carga útil SONET. A mistura realiza a operação XOR da carga útil com uma longa sequência pseudoaleatória antes de ser transmitida. O problema é que o fluxo de bits SONET precisa de transições de bits frequentes para realizar a sincronização. Essas transições vêm naturalmente com a variação em sinais de voz, mas na comunicação de dados o usuário escolhe a informação enviada e poderia enviar um pacote com uma longa sequência de 0s. Com a mistura, a probabilidade de um usuário causar problemas enviando uma longa sequência de 0s se torna extremamente baixa.

Antes que os quadros PPP possam ser transportados por linhas SONET, o enlace PPP precisa ser estabelecido e configurado. As fases pelas quais o enlace passa quando é criado, usado e removido novamente aparecem na Figura 3.18.

O enlace começa com a linha no estado *DEAD*, o que significa que não há nenhuma conexão na camada física. Depois de estabelecida a conexão na camada física, o enlace passa para a fase *ESTABLISH*. Nesse ponto, as partes do PPP trocam uma série de pacotes LCP, cada um transportado no campo *Carga útil* de um quadro PPP, para selecionar as opções do PPP para o enlace a partir das possibilidades mencionadas anteriormente. A parte que inicia propõe opções e a parte que responde as aceita ou rejeita, total ou parcialmente. A parte que responde também pode fazer propostas alternativas.

Se a negociação de opções do LCP for bem-sucedida, o enlace chega à fase *AUTHENTICATE*. Agora, as duas partes poderão verificar suas identidades mutuamente, se desejarem. Se a autenticação tiver sucesso, o estado *NETWORK* é alcançado e diversos pacotes NCP são enviados para configurar a camada de rede. É difícil generalizar a respeito dos protocolos NCP, pois cada um é específico a algum protocolo da camada de rede e permite que sejam feitas solicitações de configuração específicas a esse protocolo. Para o IP, por exemplo, a atribuição de endereços IP às duas extremidades do enlace é a possibilidade mais importante.

Quando a fase *OPEN* é alcançada, o transporte de dados pode ser feito. É nesse estado que os pacotes IP são transportados em quadros PPP pela linha SONET. Quando o transporte de dados é concluído, o enlace entra na fase *TERMINATE* e, de lá, volta ao estado *DEAD* quando a conexão da camada física é desativada.

3.5.2 ADSL (Asymmetric Digital Subscriber Line)

A ADSL conecta milhões de assinantes domésticos à Internet em taxas de Mbits/s pelo mesmo circuito terminal de telefone usado para o serviço telefônico tradicional. Na Seção 2.5.3, descrevemos como um dispositivo chamado modem DSL é colocado em uma residência. Ele envia bits

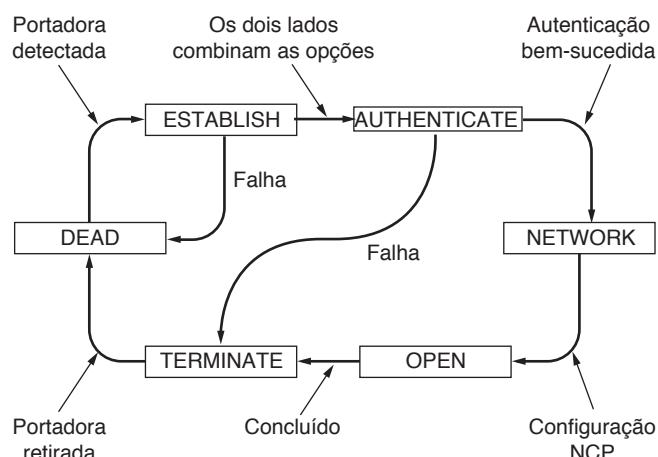



Figura 3.18 | Diagrama de estado para ativar e desativar um enlace PPP.

do circuito terminal para um dispositivo chamado DSLAM (DSL Access Multiplexer), na estação local da companhia telefônica. Agora, vamos examinar com mais detalhes como os pacotes são transportados por enlaces ADSL.

A imagem geral para os protocolos e dispositivos usados com ADSL aparece na Figura 3.19. Diferentes protocolos são empregados em diferentes redes, de modo que escolhemos mostrar o caso mais popular. Dentro da casa, um computador envia pacotes IP ao modem DSL usando uma camada de enlace como o padrão Ethernet. O modem DSL, então, envia os pacotes IP do circuito terminal para o DSLAM usando os protocolos que estamos prestes a estudar. No DSLAM (ou um roteador conectado a ele, dependendo da implementação), os pacotes IP são extraídos e entram em uma rede do ISP, de modo que possam alcançar qualquer destino na Internet.

Os protocolos mostrados pelo enlace ADSL na Figura 3.19 começam de baixo, com a camada física ADSL. Eles são baseados em um esquema de modulação digital, chamado multiplexação por divisão de frequência ortogonal (também conhecido como multitom discreto), como vimos na Seção 2.5.3. Perto do topo da pilha, logo abaixo da camada de rede IP, está o PPP. Esse protocolo é o mesmo PPP que estudamos para transportes de pacote sobre SONET. Ele funciona da mesma maneira para estabelecer e configurar o enlace e transportar pacotes IP.

Entre ADSL e PPP estão ATM e AAL5. Estes são protocolos novos, que ainda não estudamos. O **modo de transferência assíncrono**, ou **ATM** (**Asynchronous Transfer Mode**), foi elaborado no início da década de 90 e lançado com incrível entusiasmo. Ele prometia uma tecnologia de rede que resolveria os problemas de telecomunicações do mundo, reunindo voz, dados, televisão a cabo, telégrafo, pombo-correio, latinhas conectadas por barbante, tambores e tudo o mais em um sistema integrado, que poderia fazer tudo para todos. Isso não aconteceu. Em grande parte, os problemas do ATM eram semelhantes aos que já descrevemos com relação aos protocolos OSI, ou seja, má sincronização, tecnologia, implementação e política. Apesar disso, o ATM teve muito mais sucesso do que o padrão OSI. Embora não tenha ganho o mundo, ele continua sendo muito usado em nichos, incluindo linhas de acesso de banda larga, como DSL, e enlaces de WAN dentro das redes telefônicas.

O ATM é uma camada de enlace baseada na transmissão de **células** de informação de tamanho fixo. O 'assíncrono' em seu nome significa que as células nem sempre precisam ser enviadas da maneira como os bits são continuamente enviados por linhas síncronas, como SONET. As células só precisam ser enviadas quando existe informação para transportar. O ATM é uma tecnologia orientada à conexão. Cada célula transporta um identificador do **circuito virtual** em seu cabeçalho, e os dispositivos usam esse identificador para encaminhar células ao longo dos caminhos das conexões estabelecidas.

Cada célula possui 53 bytes de extensão, consistindo em uma carga útil de 48 bytes mais um cabeçalho de 5 bytes. Usando células pequenas, o ATM pode dividir a largura de banda de um enlace de camada física entre diferentes usuários, de forma flexível, em fatias finas. Esse recurso é útil quando, por exemplo, são enviados voz e dados por um enlace sem ter longos pacotes de dados, que causariam enormes atrasos nas amostras de voz. A escolha incomum para o tamanho da célula (por exemplo, em comparação com a escolha mais natural de uma potência de 2) é uma indicação de como o projeto do ATM foi político. O tamanho de 48 bytes para a carga útil foi um meio-termo para resolver um impasse entre a Europa, que queria células de 32 bytes, e os Estados Unidos, que queriam células de 64 bytes. Uma breve visão geral do ATM pode ser encontrada em Siu e Jain (1995).

Para enviar dados por uma rede ATM, eles precisam ser mapeados para uma sequência de células. Esse mapeamento é feito com uma camada de adaptação ATM em um processo chamado segmentação e remontagem. Várias camadas de adaptação foram definidas para diferentes serviços, variando de amostras periódicas de voz até pacotes de dados. A principal usada para os pacotes de dados é a **camada de adaptação ATM 5**, ou **AAL5** (**ATM Adaptation Layer 5**).

Um quadro AAL5 aparece na Figura 3.20. Em vez de um cabeçalho, ele tem um final que oferece o comprimento e um CRC de 4 bytes para detecção de erro. Naturalmente, o CRC é o mesmo usado para redes locais PPP e IEEE 802, como Ethernet. Wang e Crowcroft (1992) mostraram que ele é forte o bastante para detectar erros não tradicionais,

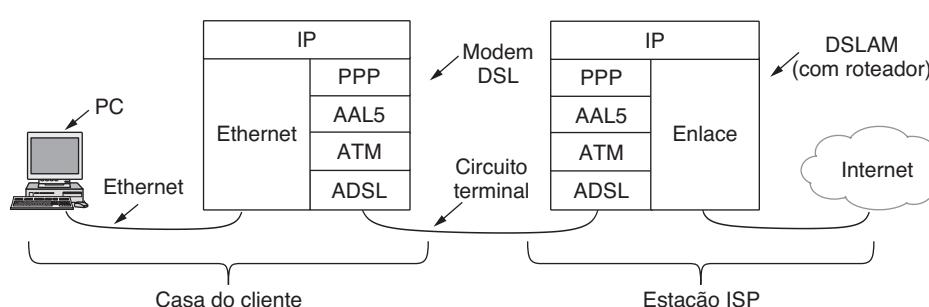



Figura 3.19 | Pilhas de protocolos ADSL.

como reordenação de célula. Assim como uma carga útil, o quadro AAL5 tem preenchimento. Isso encerra o tamanho geral como um múltiplo de 48 bytes, de modo que o quadro pode ser dividido igualmente em células. Não são necessários endereços no quadro, pois o identificador de circuito virtual transportado em cada célula o levará ao destino correto.

Agora que já descrevemos o ATM, só precisamos descrever como o PPP utiliza o ATM no caso da ADSL. Isso é feito com outro padrão, chamado **PPPoA** (**PPP over ATM**). Esse padrão não é realmente um protocolo (de modo que não aparece na Figura 3.19), e sim mais uma especificação de como trabalhar com quadros PPP e AAL5. Ele é descrito na RFC 2364 (Gross et al., 1998).

Somente os campos de protocolo e carga útil do PPP são colocados na carga útil do AAL5, como vemos na Figura 3.20. O campo de protocolo indica ao DSLAM no outro extremo se a carga útil é um pacote IP ou um pacote de outro protocolo, como LCP. O canto extremo sabe que as células contêm informações do PPP porque um circuito virtual ATM é estabelecido para essa finalidade.

Dentro do quadro AAL5, o enquadramento PPP não é necessário, pois não teria nenhuma finalidade; ATM e AAL5 já oferecem o enquadramento, e mais seria inútil. O CRC do PPP também não é necessário, pois o AAL5 já inclui o mesmo CRC. Esse mecanismo de detecção de erro suplementa a codificação da camada física da ADSL de um código de Reed-Solomon para correção de erro e um CRC de 1 byte para a detecção de quaisquer erros restantes, não descobertos de outra maneira. Esse esquema tem um mecanismo de recuperação de erros muito mais sofisticado do que quando os pacotes são enviados por uma linha SONET, pois a ADSL é um canal com muito mais ruído.

3.6 Resumo

A tarefa da camada de enlace de dados é converter o fluxo de dados brutos fornecido pela camada física em um fluxo de quadros a ser utilizado pela camada de rede. A camada de enlace pode apresentar esse fluxo com níveis de confiabilidade variados, desde o serviço não orientado à conexão sem confirmação até o serviço confiável, orientado à conexão.

Diversos métodos de enquadramento são utilizados, inclusive a contagem e a inserção de bytes e de bits. Os protocolos de enlace de dados podem oferecer recursos de controle de erro para a detecção ou correção de quadros danificados e a retransmissão de quadros perdidos. Para evitar que um transmissor rápido sobrecarregue um receptor lento, o protocolo de enlace de dados também pode fornecer controle de fluxo. O mecanismo de janela deslizante é bastante utilizado para integrar os controles de erros de fluxo de maneira simples. Quando o tamanho da janela é de 1 pacote, o protocolo é stop-and-wait.

Os códigos para correção e detecção de erro acrescentam informações redundantes às mensagens usando diversas técnicas matemáticas. Os códigos de convolução e de Reed-Solomon são muito utilizados para correção de erro, mas os códigos de verificação de paridade de baixa densidade estão ganhando popularidade. Os códigos para detecção de erro usados na prática incluem verificações de redundância cíclica e checksums. Todos esses códigos podem ser aplicados na camada de enlace, bem como na camada física e em camadas mais altas.

Examinamos diversos protocolos que oferecem uma camada de enlace confiável usando confirmações e retransmissões, ou ARQ (Automatic Repeat reQuest), sob suposições mais realistas. Começando a partir de um ambiente livre de erros, em que o receptor pode tratar de qualquer quadro que lhe é enviado, apresentamos o controle de fluxo, seguido pelo controle de erro com números de sequência e o algoritmo stop-and-wait. Depois, usamos o algoritmo de janela deslizante para permitir a comunicação bidirecional e apresentamos o conceito de piggybacking. Os dois últimos protocolos realizam o pipeline da transmissão de vários quadros para impedir que o transmissor bloqueie um enlace com um longo atraso de propagação. O receptor pode descartar todos os quadros menos o próximo na sequência, ou então armazenar em buffer os quadros fora de ordem e enviar confirmações negativas, aumentando a eficiência da largura de banda. A primeira estratégia é um protocolo go-back-n, e a segunda é um protocolo de repetição seletiva.

A Internet utiliza o PPP como protocolo de enlace de dados em linhas ponto a ponto. Ele oferece um serviço não orientado a conexões sem confirmação, usando bytes de flag para delimitar quadros e um CRC para detectar erros. Ele é usado para transportar pacotes por uma série de enlaces, incluindo enlaces SONET em redes a longas distâncias e enlaces ADSL para residências.

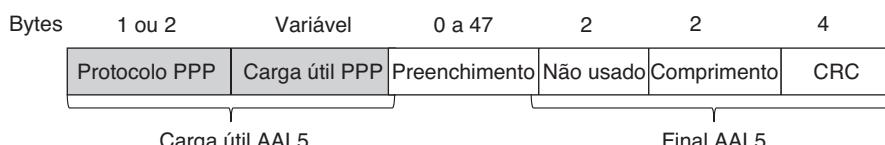



Figura 3.20 | Quadro AAL5 transportando dados PPP.

Problemas

1. Um pacote de uma camada superior está dividido em dez quadros, cada um deles com 80% de chance de chegar sem danos. Se o protocolo de enlace de dados não fizer nenhum controle de erros, quantas vezes em média a mensagem deverá ser enviada para que o processo inteiro seja concluído?
2. A codificação de caracteres a seguir é usada em um protocolo de enlace de dados:

   A: 01000111; B: 11100011; FLAG: 01111110; ESC: 11100000

   Mostre a sequência de bits transmitida (em binário) para o quadro de quatro caracteres A B ESC FLAG quando é utilizado cada um dos métodos de enquadramento a seguir:

   (a) Contagem de caracteres.

   (b) Bytes de flag com inserção de bytes.

   (c) Bytes de flag no início e no fim, com inserção de bits.
3. O fragmento de dados a seguir ocorre no meio de um fluxo de dados para o qual é usado o algoritmo de inserção de bytes descrito no texto: A B ESC C ESC FLAG FLAG D. Qual será a saída após a inserção?
4. Qual é o overhead máximo no algoritmo de inserção de bytes?
5. Um de seus colegas, Avarento, assinalou que é um desperdício encerrar cada quadro com um byte de flag e depois iniciar o próximo com um segundo byte de flag. Um único byte de flag também poderia servir, e um byte economizado é um byte ganho. Você concorda?
6. Uma sequência de bits, 0111101111101111110, precisa ser transmitida na camada de enlace de dados. Qual é a sequência realmente transmitida após a inserção de bits?
7. Você consegue imaginar alguma circunstância em que seria preferível um protocolo de loop aberto (por exemplo, um código de Hamming) aos protocolos de feedback discutidos neste capítulo?
8. Para proporcionar maior confiabilidade que a obtida com um único bit de paridade, um esquema de codificação para detecção de erro utiliza um bit de paridade para verificar todos os bits de numeração ímpar e um segundo para todos os bits de numeração par. Qual é a distância de Hamming desse código?
9. As mensagens de 16 bits são transmitidas com o uso de um código de Hamming. Quantos bits de verificação são necessários para assegurar que o receptor poderá detectar e corrigir erros de único bit? Mostre o padrão de bits transmitido no caso da mensagem 1101001100110101. Suponha que seja usada paridade par no código de Hamming.
10. Um código de Hamming de 12 bits cujo valor hexadecimal é 0xE4F chega a um receptor. Qual era o valor original em hexadecimal? Suponha que não exista mais de 1 bit com erro.
11. Uma forma de detectar erros é transmitir dados como um bloco de *n* linhas com *k* bits por linha e acrescentar bits de paridade a cada linha e a cada coluna. O canto inferior direito é um bit de paridade que verifica sua linha e sua coluna. Esse esquema detectará todos os erros simples (isolados)? E os erros duplos? E os erros triplos? Mostre que esse esquema não pode detectar alguns erros de quatro bits.
12. Suponha que sejam transmitidos dados em blocos com tamanhos de 1.000 bits. Qual é a taxa máxima de erro sob a qual o mecanismo de detecção de erro e retransmissão (1 bit de paridade por bloco) é melhor do que usar o código de Hamming? Suponha que os erros de bit sejam independentes um do outro e nenhum erro de bit ocorra durante a retransmissão.
13. Um bloco de bits com *n* linhas e *k* colunas utiliza bits de paridade horizontais e verticais para a detecção de erros. Imagine que exatamente 4 bits sejam invertidos em virtude de erros de transmissão. Derive uma expressão para a probabilidade de que o erro não seja detectado.
14. Usando o codificador convolucional da Figura 3.7, qual é a sequência de saída quando a sequência de entrada é 10101010 (da esquerda para a direita) e o estado interno é inicialmente de oito bits 0?
15. Suponha que uma mensagem 1001 1100 1010 0011 seja transmitida usando o checksum da Internet (palavra de 4 bits). Qual é o valor do checksum?
16. Qual é o resto obtido pela divisão de $x^7 + x^5 + 1$ pelo polinômio gerador $x^3 + 1$?
17. Um fluxo de bits 10011101 é transmitido com a utilização do método de CRC-padrão descrito no texto. O polinômio gerador é $x^3 + 1$. Mostre a sequência de bits real transmitida. Suponha que o terceiro bit a partir da esquerda seja invertido durante a transmissão. Mostre que esse erro é detectado na extremidade receptora. Dê um exemplo de erro de bit, na sequência de bits transmitida, que não será detectado pelo receptor.
18. Uma mensagem de 1.024 bits é enviada contendo 992 bits de dados e 32 bits de CRC. O CRC é calculado com o polinômio de CRC de grau 32, do padrão IEEE 802. Para cada um dos seguintes casos, explique se os erros durante a transmissão da mensagem serão detectados pelo receptor:

    (a) Houve um erro de bit simples.

    (b) Houve dois erros de bits isolados.

    (c) Houve 18 erros de bits isolados.

    (d) Houve 47 erros de bits isolados.

    (e) Houve um erro em rajada longa de 24 bits.

    (f) Houve um erro em rajada longa de 35 bits.
19. Na discussão do protocolo ARQ na Seção 3.3.3, esboçamos uma situação que resultou no receptor aceitando duas cópias do mesmo quadro, em decorrência de uma perda do quadro de confirmação. É possível que um receptor aceite várias cópias do mesmo quadro quando nenhum dos quadros (mensagem ou confirmação) foi perdido?
20. Um canal tem uma taxa de bits de 4 kbps e um atraso de propagação de 20 ms. Para que faixa de variação de tamanhos de quadros a técnica stop-and-wait proporciona uma eficiência de pelo menos 50%?
21. No protocolo 3, é possível que o transmissor inicialize o timer quando ele já estiver funcionando? Nesse caso, como isso poderia acontecer? Se não, por que não é possível?
22. Um tronco T1 com o comprimento de 3.000 Km é utilizado para transmitir quadros de 64 bytes usando o pro-

tocolo 5. Se a velocidade de propagação for de 6 μs/Km, quantos bits deverão ter os números de sequência?

23. Imagine que um protocolo de janela deslizante utilize tantos bits para números de sequência que nunca ocorra sobreposição. Que relações devem ser mantidas entre os quatro limites e o tamanho da janela, que é constante e idêntica para o transmissor e o receptor?

24. Se a função *between* do protocolo 5 verificasse a condição $a \leq b \leq c$ em vez da condição $a \leq b < c$, isso teria algum efeito sobre a exatidão ou a eficiência do protocolo? Explique sua resposta.

25. No protocolo 6, quando um quadro de dados chega, é feita uma verificação para confirmar se o número de sequência é diferente do esperado, e se *no_nak* é verdadeira. Se as duas condições forem verdadeiras, será enviada uma NAK. Caso contrário, o timer auxiliar será iniciado. Imagine que o comando else tenha sido omitido. Essa alteração afetaria a exatidão do protocolo?

26. Imagine que o loop while de três instruções próximo ao fim do protocolo 6 fosse removido do código. Isso afetaria a exatidão do protocolo ou apenas o desempenho? Explique sua resposta.

27. A distância entre a Terra e um planeta distante é de aproximadamente 9×10^{10} m. Qual é a utilização do canal se um protocolo stop-and-wait for usado para a transmissão de quadros em um enlace ponto a ponto de 64 Mbps? Suponha que o tamanho do quadro seja de 32 KB e a velocidade da luz seja de 3×10^8 m/s.

28. No problema anterior, considere que um protocolo de janela deslizante seja usado em seu lugar. Para qual tamanho da janela de transmissão a utilização do enlace será de 100%? Você pode ignorar os tempos de processamento do protocolo no transmissor e no receptor.

29. No protocolo 6, o código de *frame_arrival* tem uma seção utilizada para NAKs. Essa seção será chamada se o quadro recebido for uma NAK e outra condição for satisfeita. Crie uma situação em que a presença dessa outra condição seja essencial.

30. Considere a operação do protocolo 6 sobre uma linha perfeita (isto é, livre de erros) de 1 Mbps. O tamanho máximo de quadro é de 1.000 bits. Novos pacotes são gerados a cada segundo. O intervalo de timeout é de 10 ms. Se o timer especial de confirmação fosse eliminado, ocorreriam timeouts desnecessários. Quantas vezes a mensagem média seria transmitida?

31. No protocolo 6, $MAX_SEQ = 2^n - 1$. Embora essa condição seja evidentemente desejável para tornar a utilização dos bits de cabeçalho mais eficiente, não demonstramos que ela é essencial. Por exemplo, o protocolo funciona corretamente para $MAX_SEQ = 4$?

32. Quadros de 1.000 bits são enviados por um canal de 1 Mbps usando um satélite geoestacionário cujo tempo de propagação a partir da Terra é de 270 ms. As confirmações sempre são transportadas por piggyback em quadros de dados. Os cabeçalhos são muito curtos. São utilizados números de sequência de 3 bits. Qual é a utilização máxima do canal que é possível alcançar para:
    (a) Stop-and-wait?
    (b) Protocolo 5?
    (c) Protocolo 6?

33. Calcule a fração da largura de banda desperdiçada em overhead (cabeçalhos e retransmissões) para o protocolo 6 em um canal de satélite de 50 kbps bastante carregado, contendo quadros de dados com 40 bits de cabeçalho e 3.960 bits de dados. Suponha que o tempo de propagação do sinal a partir da Terra até o satélite seja de 270 ms. Os quadros ACK nunca ocorrem. Os quadros NAK têm 40 bits. A taxa de erro para os quadros de dados é de 1% e é insignificante para os quadros NAK. Os números de sequência têm 8 bits.

34. Considere um canal de satélite de 64 kbps livre de erros utilizado para enviar quadros de dados de 512 bytes em um sentido, com confirmações muito curtas voltando no outro sentido. Qual é o throughput máximo para os tamanhos de janelas iguais a 1, 7, 15 e 127? O tempo de propagação entre a Terra e o satélite é de 270 ms.

35. Um cabo com 100 km de comprimento funciona na taxa de dados T1. A velocidade de propagação no cabo é igual a 2/3 da velocidade da luz no vácuo. Quantos bits o cabo pode conter?

36. Cite pelo menos um motivo pelo qual o PPP utiliza a inserção de bytes e não a inserção de bits para evitar que bytes de flag acidentais na carga útil causem confusão.

37. Qual é o overhead mínimo para o envio de um pacote IP usando o PPP? Leve em consideração apenas o overhead introduzido pelo próprio PPP, e não o overhead do cabeçalho IP. Qual é o overhead máximo?

38. Um pacote IP de 100 bytes é transmitido por um circuito terminal usando a pilha de protocolos ADSL. Quantas células ATM serão transmitidas? Descreva seu conteúdo brevemente.

39. O objetivo deste exercício de laboratório é implementar um mecanismo de detecção de erro usando o algoritmo de CRC-padrão descrito no texto. Escreva dois programas, um *gerador* e um *verificador*. O programa *gerador* lê na entrada-padrão uma mensagem de *n* bits que tem a forma de uma sequência de valores 0 e 1 como uma linha de texto ASCII. A segunda linha é o polinômio de *k* bits, também em ASCII. A saída-padrão é uma linha de texto ASCII com $n + k$ valores, e 0 e 1 representam a mensagem a ser transmitida. Em seguida, são atribuídos valores de saída ao polinômio, exatamente como ele foi lido na entrada. O programa verificador lê a saída no programa gerador e transmite uma mensagem indicando se ela está correta ou não. Por fim, escreva um programa, chamado *alterar*, que inverta 1 bit na primeira linha, dependendo de seu argumento (o número do bit, considerando o bit mais à esquerda como igual a 1), mas copie as duas linhas restantes de forma correta. Digitando:

    ```
    gerador < arquivo | verificador
    ```

    você deverá ver que a mensagem está correta; porém, digitando

    ```
    gerador < arquivo | alterar arg | verificador
    ```

    você deverá obter a mensagem de erro.

A subcamada de controle de acesso ao meio

4 Capítulo

Os enlaces de rede podem ser divididos em duas categorias: a dos que utilizam conexões ponto a ponto e a daqueles que utilizam canais de broadcast. Estudamos os enlaces ponto a ponto no Capítulo 2; este capítulo trata das redes de broadcast e seus protocolos.

Em qualquer rede de broadcast, a questão fundamental é determinar quem tem direito de usar o canal quando há uma disputa por ele. Para tornar essa questão mais clara, considere uma chamada de conferência, na qual seis pessoas, em seis telefones, estão conectadas entre si, de forma que cada uma pode ouvir e falar com todas as outras. É muito provável que, quando uma delas parar de falar, duas ou mais comecem a falar ao mesmo tempo, causando confusão. Em uma reunião face a face, a confusão é evitada por meios externos. Por exemplo, as pessoas levantam a mão para pedir permissão para falar. Quando apenas um canal está disponível, a determinação de quem deve ser o próximo a falar é muito mais difícil. Existem vários protocolos destinados a solucionar o problema, e eles formam o conteúdo deste capítulo. Na literatura, os canais de broadcast às vezes são referidos como **canais de multiacesso** ou **canais de acesso aleatório**.

Os protocolos usados para determinar quem será o próximo em um canal de multiacesso pertencem a uma subcamada da camada de enlace de dados, chamada **MAC** (**Medium Access Control**). A subcamada MAC é especialmente importante em LANs, particularmente nas sem fios, pois o wireless é naturalmente um canal de broadcast. Em contrapartida, as WANs utilizam enlaces ponto a ponto, com exceção das redes de satélites. Como os canais de multiacesso têm uma relação muito íntima com as LANs, neste capítulo trataremos das LANs em geral, bem como de algumas questões que não fazem parte estritamente da subcamada MAC, mas o assunto principal será o controle do canal.

Tecnicamente, a subcamada MAC é a parte inferior da camada de enlace de dados e, portanto, deveríamos tê-la estudado antes de analisar todos os protocolos ponto a ponto apresentados no Capítulo 3. No entanto, para a maioria das pessoas, a compreensão de protocolos que envolvem várias partes torna-se mais fácil depois que o funcionamento dos protocolos de duas partes é bem compreendido. Por essa razão, nos desviamos um pouco da ordem de apresentação estritamente de baixo para cima.

4.1 O problema da alocação de canais

O tema central deste capítulo é definir como alocar um único canal de broadcast entre usuários concorrentes. O canal poderia ser uma parte do espectro sem fio em uma região geográfica, ou um fio isolado ou fibra óptica ao qual vários nós são conectados. Isso não importa. Nos dois casos, o canal conecta cada usuário a todos os outros e qualquer usuário que faz uso completo do canal interfere na utilização que os outros também fazem dele.

Analisaremos primeiro as limitações dos esquemas de alocação estáticos para o tráfego em rajada. Depois, mostraremos as principais premissas usadas para modelar os esquemas dinâmicos que examinaremos nas próximas seções.

4.1.1 Alocação estática de canais

A maneira tradicional de alocar um único canal, como um tronco telefônico, entre vários usuários concorrentes é dividir sua capacidade usando um dos esquemas de multiplexação que descrevemos na Seção 2.5, como FDM (Frequency Division Multiplexing). Se existem N usuários, a largura de banda é dividida em N partes do mesmo tamanho, e a cada usuário será atribuída uma parte. Como cada usuário tem uma banda de frequência particular, não há interferência entre eles. Quando existe apenas um número pequeno e constante de usuários, cada um dos quais com um fluxo constante ou uma carga de tráfego pesada, essa divisão é um mecanismo de alocação simples e eficiente. Um exemplo de uso sem fios são as estações de rádio FM. Cada estação recebe uma parte da banda de FM e a utiliza, na maior parte do tempo, para transmitir seu sinal.

No entanto, quando o número de transmissores é grande e continuamente variável, ou quando o tráfego ocorre em rajadas, a FDM apresenta alguns problemas. Se o espectro for dividido em N regiões, e menos de N usuários estiverem interessados em estabelecer comunicação no momento, uma grande parte de espectro valioso será desperdiçada e, se mais de N usuários quiserem se comunicar, alguns deles terão o acesso negado por falta de largura de banda, mesmo que alguns dos usuários aos quais uma banda de frequência foi alocada raramente transmitam ou recebam dados.

Porém, mesmo supondo que o número de usuários poderia ser, de algum modo, mantido constante em N, a

divisão de um único canal disponível em subcanais estáticos revela uma ineficiência inerente. O problema básico é que, quando alguns usuários ficam inativos, sua largura de banda é simplesmente perdida. Eles não estão utilizando essa largura de banda, e ninguém mais pode fazê-lo. Uma alocação estática não é apropriada para a maioria dos sistemas de computadores em que o tráfego de dados ocorre em rajadas (são comuns relações de 1.000:1 entre o tráfego de pico e o tráfego médio). Em consequência disso, a maioria dos canais permanecerá ociosa na maior parte do tempo.

O fraco desempenho da FDM estática pode ser facilmente visto com um simples cálculo da teoria do enfileiramento. Vamos começar com o atraso de tempo médio, T, para um canal com capacidade C bps. Consideramos que os quadros chegam aleatoriamente, com uma taxa de chegada de λ quadros/s. O comprimento de cada quadro varia, com um comprimento médio de $1/\mu$ bits. Com esses parâmetros, a taxa de serviço do canal é μC quadros/s. Pela teoria do enfileiramento, o resultado é

$$T = \frac{1}{\mu C - \lambda}$$

(Para os curiosos, esse resultado é para uma fila "M/M/1". Ele requer que a aleatoriedade dos tempos entre as chegadas e os comprimentos de quadro siga uma distribuição exponencial ou, de modo equivalente, obedeça a uma série de Poisson.)

Em nosso exemplo, se C é 100 Mbps, o comprimento do quadro médio, $1/\mu$, é 10.000 bits e a taxa de chegada de quadros, λ, é 5.000 quadros/s, então $T = 200$ µs. Observe que, se ignorarmos o atraso de enfileiramento e simplesmente perguntarmos quanto tempo é necessário para enviar um quadro de 10.000 bits em uma rede de 100 Mbps, obteremos a resposta (incorreta) de 100 µs. Esse resultado só é valido quando não há disputa pelo canal.

Agora, vamos dividir o único canal em N subcanais independentes, cada um com capacidade de C/N bps. A taxa média de entrada em cada um dos subcanais será, agora, λ/N. Ao recalcularmos T, obteremos:

$$T_N = \frac{1}{\mu(C/N) - (\lambda/N)} = \frac{N}{\mu C - \lambda} = NT \qquad (4.1)$$

O atraso médio para o canal dividido é N vezes pior do que seria se todos os quadros estivessem, de alguma forma mágica, distribuídos de maneira ordenada em uma grande fila central. Esse mesmo resultado explica por que um banco cheio de caixas eletrônicos funciona melhor com uma única fila alimentando todas as máquinas do que uma fila separada à frente de cada máquina.

Os mesmos argumentos que se aplicam à FDM também se aplicam a outras formas de dividir o canal estaticamente. Se usássemos a multiplexação por divisão de tempo, ou TDM (Time Division Multiplexing), e alocássemos cada usuário a cada n-ésimo slot de tempo, e ainda se um usuário não empregar o slot alocado, este será simplesmente desperdiçado. O mesmo é válido se dividirmos as redes fisicamente. Usando mais uma vez nosso exemplo anterior, se substituíssemos a rede de 100 Mbps por dez redes de 10 Mbps e fizéssemos a alocação estática de cada usuário a uma delas, o atraso médio saltaria de 200 µs para 2 ms.

Como nenhum dos métodos estáticos tradicionais de alocação de canais funciona bem com um tráfego em rajada, agora trataremos dos métodos dinâmicos.

4.1.2 Premissas para a alocação dinâmica de canais

Antes de começarmos a descrever o primeiro dos muitos métodos de alocação de canais a ser discutidos neste capítulo, vale a pena formular cuidadosamente o problema da alocação. Existem cinco premissas fundamentais subjacentes a todo trabalho realizado nessa área, que serão descritas a seguir.

1. **Tráfego independente**. O modelo consiste em N **estações** independentes (computadores, telefones), cada qual com um programa ou usuário que gera quadros para transmissão. O número esperado de quadros gerados em um intervalo de duração Δt é $\lambda \Delta t$, onde λ é uma constante (a taxa de chegada de novos quadros). Uma vez gerado um quadro, a estação é bloqueada e nada faz até que o quadro tenha sido transmitido com êxito.
2. **Premissa de canal único**. Um único canal está disponível para todas as comunicações. Todas as estações podem transmitir e receber por ele. As estações são consideradas igualmente capazes, embora os protocolos possam atribuir diferentes papéis (por exemplo, prioridades) a elas.
3. **Colisões observáveis**. Se dois quadros são transmitidos simultaneamente, eles se sobrepõem no tempo, e o sinal resultante é adulterado. Esse evento é denominado **colisão**. Todas as estações podem detectar colisões. Um quadro que tenha sofrido colisão terá de ser retransmitido posteriormente. Não há outros erros além dos gerados por colisões.
4. **Tempo contínuo ou segmentado (slotted).** O tempo pode ser considerado contínuo, caso em que a transmissão do quadro pode começar a qualquer instante. Como alternativa, o tempo pode ser segmentado ou dividido em intervalos discretos (slots). As transmissões de quadros sempre começam no início de um slot. Um slot pode conter 0, 1 ou mais quadros, correspondentes a um slot ocioso, a uma transmissão bem-sucedida ou a uma colisão, respectivamente.

5. **Detecção de portadora (carrier sense) ou sem detecção de portadora**. Com a premissa de detecção de portadora, as estações conseguem detectar se o canal está sendo usado antes de tentar utilizá-lo. Se for detectado que o canal está ocupado, nenhuma estação tentará usá-lo até que ele fique livre. Se não houver detecção de portadora, as estações não conseguem detectar o canal antes de tentar utilizá-lo. Elas simplesmente vão em frente e transmitem. Somente mais tarde conseguem determinar se a transmissão foi ou não bem-sucedida.

Ainda é necessário discutir essas premissas um pouco mais. A primeira diz que as chegadas de quadro são independentes, entre estações ou em uma estação específica, e que os quadros são gerados de modo imprevisível, mas a uma taxa constante. Na realidade, essa premissa não é um modelo de tráfego de rede particularmente bom, pois sabemos que os pacotes chegam em rajadas por um intervalo de escalas de tempo (Paxson e Floyd, 1995; e Leland et al., 1994). Apesar disso, **modelos de Poisson**, como normalmente são chamados, são úteis porque são matematicamente tratáveis. Eles nos ajudam a analisar protocolos para entender, em linhas gerais, como o desempenho muda ao longo de um intervalo de operação e como ele se compara com outros projetos.

A premissa de canal único é o núcleo do modelo. Não existem formas externas de comunicação. As estações não podem levantar as mãos para solicitar que o professor lhes permita falar, de modo que precisaremos de soluções melhores.

As três premissas restantes dependem da engenharia do sistema, e diremos quais premissas são mantidas quando examinarmos um protocolo em particular.

A premissa de colisão também é básica. As estações precisam de um modo de detectar colisões se tiverem de transmitir quadros em vez de deixar que se percam. Para canais com fio, o hardware do nó pode ser projetado para detectar colisões quando elas ocorrerem. As estações podem, então, terminar suas transmissões prematuramente para evitar desperdiçar capacidade. Essa detecção é muito mais difícil para canais sem fio, de modo que as colisões normalmente são deduzidas após o fato, pela falta de um quadro de confirmação esperado. Também é possível que alguns quadros envolvidos em uma colisão sejam recebidos com sucesso, dependendo dos detalhes dos sinais e do hardware de recepção. Contudo, essa situação não é o caso comum, de modo que iremos supor que todos os quadros envolvidos em uma colisão são perdidos. Também veremos protocolos projetados para impedir colisões em primeiro lugar.

O objetivos da duas premissas alternativas em relação ao tempo é que o tempo segmentado pode ser usado para melhorar o desempenho. Porém, é preciso que as estações sigam um relógio mestre ou sincronizem suas ações entre si para dividir o tempo em intervalos distintos. Logo, ele nem sempre está disponível. Discutiremos e analisaremos os sistemas com os dois tipos de tempo. Para determinado sistema, somente um deles é válido.

Da mesma forma, uma rede pode ter a detecção de portadora ou não. Em geral, as redes com fio têm detecção de portadora. No entanto, as redes sem fio não podem usá-la de forma efetiva, porque nem toda estação pode estar dentro da faixa de rádio das outras. De modo semelhante, a detecção de portadora não estará disponível em outros ambientes nos quais uma estação não pode se comunicar diretamente com outras estações, por exemplo, um modem a cabo no qual as estações precisam se comunicar pelo headend do cabo. Observe que a palavra "portadora" (*carrier*), nesse sentido, se refere ao sinal elétrico enviado pelo cabo, e não tem nenhuma relação com concessionárias de comunicações (*common carriers*) — por exemplo, as empresas de telefonia — que remontam à época do Pony Express.

Para evitar qualquer mal-entendido, vale a pena observar que nenhum protocolo de multiacesso garante entrega confiável. Mesmo na ausência de colisões, o receptor pode ter copiado parte do quadro incorretamente por diversos motivos. Outras partes da camada de enlace ou de camadas superiores oferecem confiabilidade.

4.2 Protocolos de acesso múltiplo

Existem muitos algoritmos conhecidos para alocar um canal de acesso múltiplo. Nas seções a seguir, estudaremos uma pequena amostra dos mais interessantes e apresentaremos alguns exemplos de sua utilização.

4.2.1 ALOHA

A história do nosso primeiro protocolo de acesso múltiplo, ou MAC, começa no Havaí primitivo, no início da década de 70. Nesse caso, 'primitivo' pode ser interpretado como 'não tendo um sistema telefônico funcional'. Isso não tornava a vida mais agradável para o pesquisador Norman Abramson e seus colegas da Universidade do Havaí, que estavam tentando conectar usuários nas ilhas remotas ao computador principal em Honolulu. Esticar seus próprios cabos sob o Oceano Pacífico estava fora de cogitação e, portanto, eles procuravam uma solução diferente.

A solução encontrada usava rádios de curta distância, com cada terminal de usuário compartilhando a mesma frequência upstream para enviar quadros ao computador central. Isso incluía um método simples e elegante para resolver o problema de alocação de canal. Seu trabalho foi ampliado por vários pesquisadores desde então (Schwartz e Abramson, 2009). Embora o trabalho de Abramson, denominado sistema ALOHA, usasse a radiofrequência terrestre, a ideia básica é aplicável a qualquer sistema em que

usuários sem nenhuma coordenação estão competindo pelo uso de um único canal compartilhado.

Descreveremos aqui duas versões do ALOHA: original e slotted. Elas diferem quanto ao fato de o tempo ser contínuo, como na versão original, ou dividido em slots discretos em que todos os quadros devem se encaixar.

ALOHA original

A ideia básica de um sistema ALOHA é simples: permitir que os usuários transmitam sempre que tiverem dados para enviar. Naturalmente, haverá colisões, e os quadros que colidirem serão danificados. Os transmissores precisam, de alguma maneira, descobrir se isso acontece. No sistema ALOHA, após cada estação ter transmitido seu quadro para o computador central, este computador retransmite o quadro para todas as estações. Desse modo, uma estação transmissora pode escutar por broadcast a partir do hub para ver se seu quadro passou. Em outros sistemas, como nas LANs com fios, o transmissor precisa ser capaz de escutar colisões enquanto transmite.

Se o quadro foi destruído, o transmissor apenas espera um período aleatório e o envia novamente. O tempo de espera deve ser aleatório, caso contrário os mesmos quadros continuarão a colidir repetidas vezes, de forma inflexível. Os sistemas em que vários usuários compartilham um canal comum de forma que possa gerar conflitos em geral são conhecidos como sistemas de **disputa**.

A Figura 4.1 mostra um esboço da geração de quadros em um sistema ALOHA. Os quadros foram criados com o mesmo comprimento porque o throughput dos sistemas ALOHA é maximizado quando o comprimento dos quadros é uniforme em vez de variável.

Sempre que dois quadros tentarem ocupar o canal ao mesmo tempo, haverá uma colisão e ambos serão danificados. Se o primeiro bit de um novo quadro se sobrepuser apenas ao último bit de um quadro quase terminado, os dois quadros serão totalmente destruídos (ou seja, terão checksums incorretos) e terão de ser retransmitidos posteriormente. O checksum não consegue (e não deve) fazer distinção entre uma perda total e uma perda parcial. Quadro com erro é quadro com erro, sem distinções.

Uma questão interessante é: qual é a eficiência de um canal ALOHA? Em outras palavras, que fração de todos os quadros transmitidos escapa de colisões nessas circunstâncias tão confusas? Vamos considerar primeiro um conjunto infinito de usuários interativos em seus terminais (estações). O usuário sempre se encontra em um de dois estados: digitação ou espera. Inicialmente, todos os usuários estão no estado de digitação. Quando uma linha é conectada, o usuário para de digitar e espera uma resposta. Então, a estação transmite um quadro contendo a linha e verifica o canal para saber se a transmissão foi bem-sucedida. Em caso afirmativo, o usuário vê a resposta e volta a digitar. Caso contrário, ele continua a esperar e o quadro é retransmitido continuamente até ser enviado com êxito.

O 'tempo de quadro' representa o período necessário para transmitir o quadro-padrão de comprimento fixo (isto é, o comprimento do quadro dividido pela taxa de bits). Nesse ponto, supomos que os novos quadros sejam gerados pelas estações de acordo com uma distribuição de Poisson, com uma média de N quadros por tempo de quadro. (A premissa de população infinita é necessária para garantir que N não diminuirá à medida que os usuários forem bloqueados). Se $N > 1$, a comunidade de usuários está gerando quadros em uma taxa superior à capacidade do canal, e praticamente todos os quadros sofrerão colisões. Para um throughput razoável, esperaríamos $0 < N < 1$.

Além dos novos quadros, as estações também geram retransmissões de quadros que sofreram colisões anteriormente. Vamos supor ainda que os quadros antigos e novos combinados também sejam uma distribuição de Poisson, com média G por tempo de quadro. Evidentemente, $G \geq N$. Em situações de carga baixa (ou seja, $N \approx 0$), ocorrerão poucas colisões e, portanto, haverá poucas retransmissões. Por conseguinte, $G \approx N$. Em situações de carga alta, ocorrerão várias colisões e, portanto, $G > N$. Para qualquer carga, o throughput S é simplesmente a carga oferecida, G, multiplicada pela probabilidade P_0 de uma transmissão ser

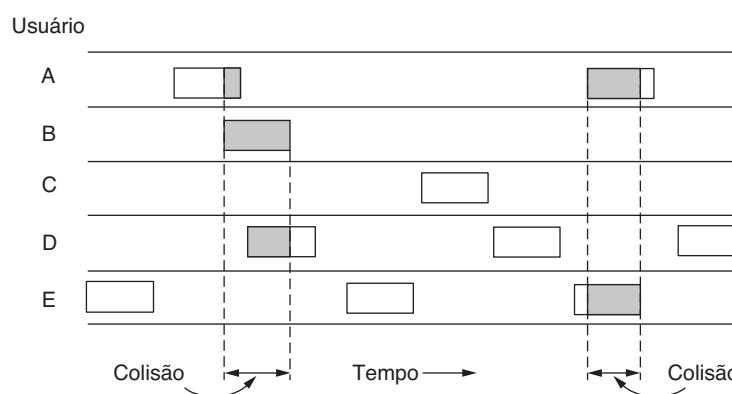



Figura 4.1 | No ALOHA original, os quadros são transmitidos em tempos totalmente arbitrários.

bem-sucedida — isto é, $S = GP_0$, onde P_0 é a probabilidade de um quadro não sofrer colisão.

Um quadro não sofrerá colisão se nenhum outro for enviado dentro de um tempo de quadro a partir de seu início, como mostra a Figura 4.2. Em que condições o quadro sombreado chegará sem erros? Seja t o tempo necessário para enviar um quadro. Se qualquer outro usuário tiver gerado um quadro no intervalo entre t_0 e $t_0 + t$, o final desse quadro colidirá com o início do quadro sombreado. Na verdade, esse quadro já estava condenado antes de o primeiro bit ser transmitido; porém, como no ALOHA original uma estação não escuta o canal antes de transmitir, não há como saber se já havia outro quadro a caminho. Da mesma forma, qualquer outro quadro iniciado entre $t_0 + t$ e $t_0 + 2t$ colidirá com o final do quadro sombreado.

A probabilidade de k quadros serem gerados durante determinado tempo de quadro, no qual G quadros são esperados, é obtida pela distribuição de Poisson

$$\Pr[k] = \frac{G^k \, e^{-G}}{k!} \qquad (4.2)$$

e, portanto, a probabilidade de zero quadros é simplesmente e^{-G}. Em um intervalo com duração de dois tempos de quadro, o número médio de quadros gerados é $2G$. A probabilidade de nenhum outro quadro ser iniciado durante todo o período de vulnerabilidade é, portanto, indicada por $P_0 = e^{-2G}$. Usando $S = GP_0$, obtemos:

$$S = Ge^{-2G}$$

A Figura 4.3 mostra a relação entre o tráfego oferecido e o throughput. O throughput máximo ocorre em $G = 0{,}5$, com $S = 1/2e$, o que corresponde aproximadamente a 0,184. Em outras palavras, o melhor que podemos esperar é uma utilização de canal de 18 por cento. Esse resultado não é muito animador, mas, com todas as pessoas transmitindo à vontade, dificilmente poderíamos esperar uma taxa de 100 por cento de êxito.

Slotted ALOHA

Logo depois que o ALOHA entrou em cena, Roberts (1972) publicou um método para duplicar a capacidade de um sistema ALOHA. Sua proposta era dividir o tempo em intervalos discretos, chamados **slots**, com cada intervalo correspondendo a um quadro. Esse método exige que os usuários concordem em relação às fronteiras dos slots. Uma forma de alcançar a sincronização entre os usuários seria ter uma estação especial que emitisse um sinal sonoro no início de cada intervalo, como um relógio.

No método de Roberts, que passou a ser conhecido como **slotted ALOHA** —, em contraste com o ALOHA original de Abramson — um computador não tem permissão para transmitir sempre que o usuário digita uma linha. Em vez disso, é necessário esperar o início do próximo slot. Consequentemente, o ALOHA original contínuo transforma-se em um sistema discreto. O período de vulnerabilidade está agora reduzido à metade. Para verificar isso, examine a Figura 4.3 e imagine as colisões que agora são possíveis. A probabilidade de não haver outro tráfego durante o mesmo slot de nosso quadro de teste é e^{-G}, o que nos leva a:

$$S = Ge^{-G} \qquad (4.3)$$

Como podemos ver na Figura 4.3, a taxa máxima do slotted ALOHA é $G = 1$, com um throughput $S = 1/e$, ou aproximadamente 0,368, o dobro do ALOHA original. Se o sistema estiver funcionando a uma taxa de $G = 1$, a probabilidade de um slot vazio será 0,368 (pela Equação 4.2). O melhor que podemos esperar com a utilização de um slotted ALOHA é 37 por cento de slots vazios, 37 por cento de sucessos e 26 por cento de colisões. O funcionamento em valores superiores de G reduz o número de slots vazios, mas aumenta exponencialmente o número de colisões. Para ver como ocorre esse rápido crescimento de colisões com G, considere a transmissão de um quadro de teste. A probabilidade de ele evitar uma colisão é

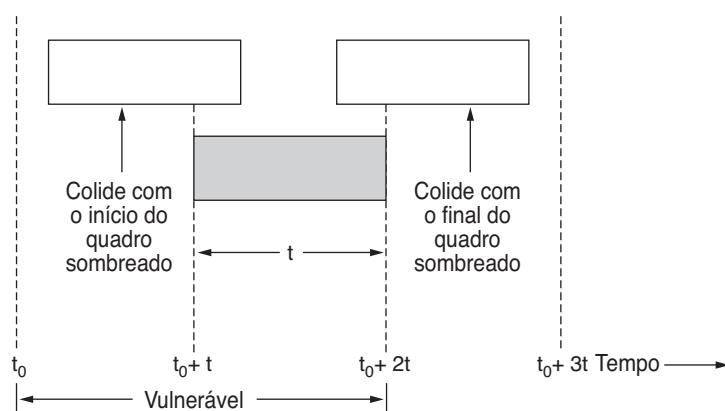



Figura 4.2 | Período de vulnerabilidade do quadro sombreado.

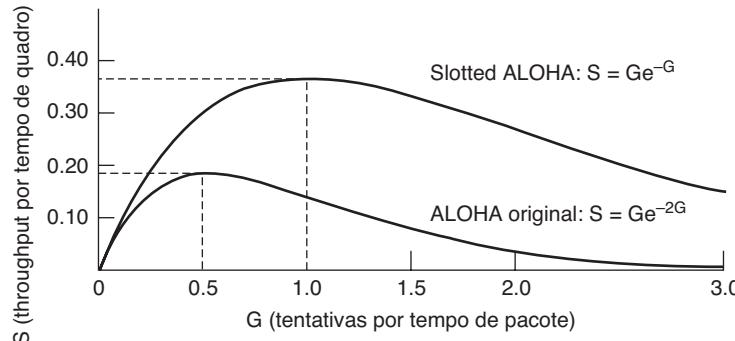



Figura 4.3 | Throughput em comparação com o tráfego oferecido para sistemas ALOHA.

de e^{-G}, que é a probabilidade de todos os outros usuários estarem inativos nesse slot. A probabilidade de uma colisão é, então, apenas $1 - e^{-G}$. A probabilidade de uma transmissão exigir exatamente k tentativas (ou seja, $k - 1$ colisões seguidas por uma transmissão bem-sucedida) é

$$P_k = e^{-G}\,(1 - e^{-G})^{k-1}$$

O número esperado de transmissões, E, por cada linha digitada em um terminal é, portanto,

$$E = \sum_{k=1}^{\infty} kP_k = \sum_{k=1}^{\infty} ke^{-G}(1-e^{-G})^{k-1} = e^{G}$$

Como resultado da dependência exponencial de E em relação a G, pequenos aumentos na carga do canal podem reduzir drasticamente seu desempenho.

O slotted ALOHA é importante por uma razão que, em princípio, talvez não seja óbvia. Ele foi criado na década de 70, foi usado em alguns sistemas experimentais e depois foi quase esquecido. Quando foi criado o acesso à Internet por cabo, surgiu o problema de como alocar um canal compartilhado entre vários usuários concorrentes, e o slotted ALOHA foi resgatado para salvar a situação. Posteriormente, várias etiquetas de RFID falando com o mesmo leitor de RFID ocasionaram outra variação do mesmo problema. O slotted ALOHA, com um punhado de outras ideias misturadas, novamente veio ao socorro. Com frequência, protocolos perfeitamente válidos caem em desuso por razões políticas (por exemplo, quando alguma grande empresa deseja que todas as outras sigam seu modelo) ou em virtude de tendências tecnológicas em constante mudança. Então, anos depois, alguém inteligente percebe que um protocolo descartado muito antes resolve seu problema atual. Por essa razão, neste capítulo estudaremos diversos protocolos elegantes que não são muito utilizados hoje, mas que poderiam facilmente ser empregados em aplicações futuras, desde que projetistas de redes em números suficientes tivessem conhecimento deles. É claro que também estudaremos muitos protocolos bastante usados atualmente.

4.2.2 Protocolos de acesso múltiplo com detecção de portadora

Com o slotted ALOHA, a melhor utilização de canal que é possível conseguir é $1/e$. Isso não surpreende, pois, com as estações transmitindo à vontade, sem prestar atenção ao que as outras estão fazendo, é provável que ocorram muitas colisões. Porém, em LANs, as estações podem detectar o que outras estão fazendo e, então, adaptam seu comportamento de acordo com essa situação. Essas redes podem atingir uma utilização melhor que $1/e$. Nesta seção, estudaremos alguns protocolos que melhoram o desempenho da rede.

Os protocolos nos quais as estações escutam uma portadora (isto é, uma transmissão) e funcionam de acordo com ela são denominados **protocolos com detecção de portadora**. Muitos deles têm sido propostos e já há muito tempo foram analisados com detalhes. Por exemplo, consulte Kleinrock e Tobagi (1975). A seguir, mencionaremos algumas versões dos protocolos com detecção de portadora.

CSMA persistente e não persistente

O primeiro protocolo com detecção de portadora que estudaremos aqui denomina-se **CSMA** (**Carrier Sense Multiple Access**) **1-persistente**. Esse é um nome extenso para indicar o esquema CSMA mais simples. Quando uma estação tem dados a transmitir, primeiro ela escuta o canal para ver se mais alguém está transmitindo no momento. Se o canal estiver desocupado, as estações enviam seus dados. Caso contrário, se o canal estiver ocupado, a estação espera até que ele fique desocupado. Então, a estação transmite um quadro. Se ocorrer uma colisão, a estação espera um intervalo de tempo aleatório e começa tudo de novo. Esse protocolo é denominado 1-persistente porque a estação transmite com probabilidade 1 sempre que encontra o canal desocupado.

Você poderia esperar que esse esquema evitasse colisões, exceto no caso raro de transmissões simultâneas, mas na verdade isso não acontece. Se duas estações estão prontas no meio da transmissão de uma terceira estação, ambas

esperarão educadamente até que a transmissão termine e, depois, ambas começarão a transmitir simultaneamente, resultando em uma colisão. Se elas não fossem tão impacientes, haveria menos colisões.

De modo mais sutil, o atraso de propagação tem um efeito importante sobre as colisões. Há uma chance de que, logo após uma estação começar a transmitir, outra estação fique pronta para transmitir e escutar o canal. Se o sinal da primeira estação ainda não tiver atingido a segunda, esta detectará um canal desocupado e também começará a transmitir, resultando em uma colisão. Essa probabilidade depende do número de quadros que cabem no canal, ou o **produto largura de banda-atraso** do canal. Se apenas uma pequena fração do quadro couber no canal, o que é o caso na maioria das LANs, uma vez que o atraso de propagação é pequeno, o risco de uma colisão acontecer é pequeno. Quanto maior o produto largura de banda-atraso, maior será a importância desse efeito e pior será o desempenho do protocolo.

Mesmo assim, esse protocolo tem um desempenho bem melhor que o ALOHA original, pois ambas as estações respeitam a transmissão e desistem de interferir no quadro de uma terceira estação. Exatamente o mesmo se aplica ao slotted ALOHA.

Um segundo protocolo com detecção de portadora é o **CSMA não persistente**. Nesse protocolo, é feita uma tentativa consciente de ser menos ávido que no protocolo anterior. Antes de transmitir, a estação escuta o canal e, se ninguém mais estiver transmitindo, inicia a transmissão. No entanto, se o canal já estiver sendo utilizado, a estação não permanecerá escutando continuamente a fim de se apoderar de imediato do canal após detectar o fim da transmissão anterior. Em vez disso, a estação aguardará durante um intervalo aleatório e, em seguida, repetirá o algoritmo. Consequentemente, esse algoritmo leva a uma melhor utilização do canal, e a atrasos maiores do que no CSMA 1-persistente.

O último protocolo é o **CSMA p-persistente**. Ele se aplica a canais segmentados (slotted) e funciona da forma apresentada a seguir. Quando está pronta para transmitir, a estação escuta o canal. Se ele estiver desocupado, a estação transmite com uma probabilidade p. Com uma probabilidade $q = 1 - p$, haverá um adiamento até o próximo slot. Se este também estiver desocupado, haverá uma transmissão ou um novo adiamento, com probabilidades p e q. Esse processo se repete até o quadro ser transmitido ou até que outra estação tenha iniciado uma transmissão. Neste último caso, ela age como se tivesse ocorrido uma colisão (ou seja, aguarda durante um intervalo aleatório e reinicia a transmissão). Se inicialmente detectar que o canal está ocupado, a estação espera pelo próximo slot e aplica o algoritmo anterior. O IEEE 802.11 usa uma melhoria do CSMA p-persistente, que discutiremos na Seção 4.4.

A Figura 4.4 mostra o throughput calculado em comparação com o tráfego oferecido para todos os três protocolos, bem como para o ALOHA original e o slotted ALOHA.

CSMA com detecção de colisões

Os protocolos CSMA persistentes e não persistentes claramente são um avanço em relação ao ALOHA, pois garantem que nenhuma estação começará a transmitir ao perceber que o canal está ocupado. Porém, se duas estações perceberem que o canal está desocupado e começarem a transmitir simultaneamente, seus sinais ainda causarão colisão. Outro avanço é que as estações podem detectar a colisão rapidamente e interromper a transmissão de forma abrupta (em vez de completá-la), pois não têm como reparar a situação. Essa estratégia economiza tempo e largura de banda.

Esse protocolo, conhecido como **CSMA/CD** (**CSMA with Collision Detection**), é a base da LAN Ethernet clássica; assim, vale a pena dedicarmos algum tempo a examiná-lo em detalhes. É importante observar que a detecção de colisão é um processo analógico. O hardware da

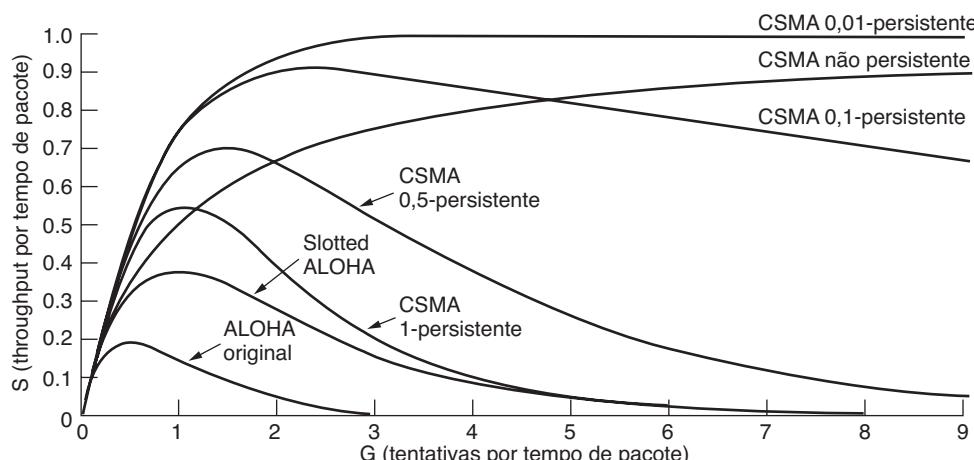



Figura 4.4 | Comparação entre a utilização do canal e a carga de vários protocolos de acesso aleatório.

estação deve escutar o canal enquanto está transmitindo. Se o sinal que ela lê de volta for diferente do sinal que está enviando, ela saberá que está havendo uma colisão. As implicações são que um sinal recebido não deverá ser pequeno em comparação com o sinal transmitido (o que é difícil para as redes sem fio, pois os sinais recebidos podem ser um milhão de vezes mais fracos do que os sinais transmitidos) e que a modulação deve ser escolhida para permitir que as colisões sejam detectadas (por exemplo, uma colisão de dois sinais de 0 volt pode muito bem ser impossível de detectar).

O CSMA/CD e vários outros protocolos de LANs utilizam o modelo conceitual apresentado na Figura 4.5. No ponto marcado com t_0, uma estação terminou a transmissão de seu quadro. Qualquer outra estação que tenha um quadro a ser enviado pode transmiti-lo. Se duas ou mais estações decidirem transmitir simultaneamente, haverá uma colisão. Se uma estação detecta uma colisão, ela cancela sua transmissão, espera um intervalo aleatório e, em seguida, tenta novamente (supondo que nenhuma outra estação tenha começado a transmitir nesse ínterim). Dessa forma, nosso modelo de CSMA/CD consistirá em períodos alternados de disputa e de transmissão, com a ocorrência de períodos de inatividade quando todas as estações estiverem em repouso (por exemplo, por falta de trabalho).

Agora, vamos analisar mais de perto os detalhes do algoritmo de disputa. Suponha que duas estações comecem uma transmissão no instante exato t_0. Quanto tempo elas levarão para perceber que houve uma colisão? A resposta a essa pergunta é essencial para determinar a duração do intervalo de disputa e, portanto, quais serão o atraso e o throughput.

O tempo mínimo para a detecção de uma colisão é apenas o tempo que o sinal leva para se propagar de uma estação até a outra. Com base nesse raciocínio, você poderia pensar que uma estação que não detectasse uma colisão durante um intervalo igual ao tempo de propagação em todo o cabo, após ter iniciado sua transmissão, teria certeza de haver se apoderado do cabo. Com o termo "apoderado", queremos dizer que todas as outras estações sabem da transmissão e não interferirão. Essa conclusão está incorreta.

Considere a pior hipótese possível a seguir. Seja τ o tempo de propagação de um sinal entre as duas estações mais distantes. Em t_0, uma estação começa a transmitir. Em $t_0 + \tau - \varepsilon$, um instante antes de o sinal chegar à estação mais distante, essa estação também começa a transmitir. É claro que ela detecta a colisão quase instantaneamente e para, mas o pequeno ruído causado pela colisão não retorna à estação original até o período $2\tau - \varepsilon$. Em outras palavras, na pior das hipóteses, uma estação só poderá ter certeza de ter se apoderado do canal após transmitir durante o período 2τ sem escutar uma colisão.

Compreendendo isso, podemos pensar na disputa do CSMA/CD como um sistema slotted ALOHA, com uma largura de slot igual a 2τ. Em um cabo coaxial de 1 km de comprimento, $\tau \approx 5\mu s$. A diferença para CSMA/CD em comparação com o slotted ALOHA é que os slots em que apenas uma estação transmite (ou seja, em que o canal é apoderado) são acompanhados pelo restante de um quadro. Essa diferença melhorará bastante o desempenho se o tempo do quadro for muito maior que o tempo de propagação.

4.2.3 Protocolos livres de colisão

Embora as colisões não ocorram com o CSMA/CD depois que uma estação captura o canal sem ambiguidade, elas ainda podem ocorrer durante o período de disputa. Essas colisões afetam de modo adverso o desempenho do sistema, em especial quando o cabo é longo (ou seja, quando τ é grande) e os quadros são curtos. As colisões não só reduzem a largura de banda, mas também tornam variável o tempo para transmitir um quadro, o que não é bom para um tráfego em tempo real, como VoIP. Além disso, o CSMA/CD não é aplicável de maneira universal.

Nesta seção, examinaremos alguns protocolos que resolvem a disputa pelo canal sem a ocorrência de colisões, nem mesmo durante o período de disputa. A maioria desses protocolos não é usada atualmente em sistemas importantes, mas, em um campo que muda rapidamente, a existência de alguns protocolos com excelentes propriedades disponíveis para sistemas futuros frequentemente é algo bom.

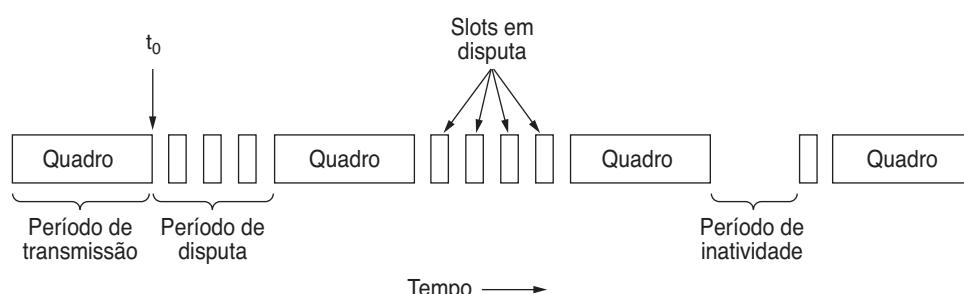



Figura 4.5 | O CSMA/CD pode estar em um destes três estados: disputa, transmissão ou inatividade.

Nos protocolos que descreveremos, vamos supor que existam exatamente N estações, cada uma programada com um endereço exclusivo de 0 até $N - 1$. O fato de que algumas estações talvez possam estar inativas durante parte do tempo não tem importância. Também supomos que o atraso de propagação é desprezível. A pergunta básica permanece: que estação terá a posse do canal após uma transmissão bem-sucedida? Continuaremos a utilizar o modelo mostrado na Figura 4.5 com seus slots discretos de disputa.

O protocolo bit-map

Em nosso primeiro protocolo livre de colisão, o **método básico bit-map**, cada período de disputa consiste em exatamente N slots. Se tiver um quadro para transmitir, a estação 0 envia um bit 1 durante o slot número zero. Nenhuma outra estação poderá transmitir durante esse slot. Independentemente do que a estação 0 fizer, a estação 1 terá a oportunidade de transmitir um bit 1 durante o slot 1, mas apenas se tiver um quadro na fila para ser enviado. Em geral, é possível que a estação j informe que tem um quadro para transmitir inserindo um bit 1 no slot j. Depois que todos os N slots tiverem passado, cada estação terá total conhecimento de quais estações desejam transmitir. Nesse ponto, elas começam a transmitir em ordem numérica (ver Figura 4.6).

Como todas as estações concordam sobre quem será a próxima a transmitir, nunca haverá colisões. Após a última estação ter transmitido seu quadro, um evento que todas as estações podem monitorar com facilidade, inicia-se outro período de disputa de N bits. Se uma estação ficar pronta logo após seu slot de bits ter passado, ela não conseguirá transmitir e precisará permanecer inativa até que todas as outras estações tenham tido a chance de transmitir e o bit-map tenha voltado a passar por ela.

Protocolos como esse, nos quais o desejo de transmitir é difundido antes de ocorrer a transmissão real, são chamados **protocolos de reserva**, pois eles reservam a propriedade do canal com antecedência e impedem colisões. Vamos analisar rapidamente o desempenho desse protocolo. Para facilitar, mediremos o tempo em unidades do slot de bits de disputa, com os quadros de dados consistindo em d unidades de tempo.

Em condições de carga baixa, o bit-map simplesmente será repetido várias vezes, por falta de quadros de dados. Considere a situação do ponto de vista de uma estação com numeração baixa, como 0 ou 1. Normalmente, quando ela fica pronta para enviar, o slot 'atual' estará em algum ponto no meio do bit-map. Em média, a estação terá de esperar $N/2$ slots para que a varredura atual seja concluída e mais N slots completos até que a varredura seguinte se encerre, para poder começar a transmitir.

As estações que estiverem aguardando e tiverem números mais altos obterão resultados melhores. Em geral, essas estações só precisarão esperar pela metade de uma varredura ($N/2$ slots de bits) antes de iniciar a transmissão. As estações com numeração mais alta raramente precisam esperar pela próxima varredura. Como as estações de numeração baixa precisam esperar em média 1,5 N slot e as de numeração alta precisam esperar em média 0,5 N slot, a média para todas as estações é N slots.

É fácil calcular a eficiência do canal em carga baixa. O overhead por quadro é de N bits, e o volume de dados é de d bits, o que resulta em uma eficiência igual a $d/(d + N)$.

Sob carga alta, quando todas as estações têm algo a enviar o tempo todo, o período de disputa de N bits é dividido proporcionalmente entre N quadros, produzindo um overhead de apenas 1 bit por quadro, ou uma eficiência igual a $d/(d + 1)$. O atraso médio para um quadro é equivalente à soma do tempo de espera na fila dentro da estação, mais um adicional de $(N - 1)d + N$, uma vez que ele alcança o início de sua fila interna. Esse intervalo é o tempo necessário para esperar até que todas as outras estações tenham sua vez para enviar um quadro e outro bit-map.

Passagem de tokens

A essência do protocolo bit-map é que ele permite que cada estação transmita um quadro por vez, em uma ordem predefinida. Outra forma de realizar a mesma coisa é passar uma pequena mensagem, chamada **token** ou **sinal**, de uma estação para a seguinte, na mesma ordem predefinida. O token representa a permissão para enviar. Se uma estação tem um quadro na fila para transmissão quando recebe o token, ela pode enviar esse quadro antes de passar o token para a próxima estação. Se ela não tiver um quadro na fila, ela simplesmente passará o token.

Em um protocolo que utiliza a topologia de anel de tokens (**token ring**), a topologia da rede em anel é usada para definir a ordem em que as estações transmitem. As estações são conectadas às seguintes formando um anel único. A passagem do token para a estação seguinte consiste simplesmente em receber o token em uma direção e

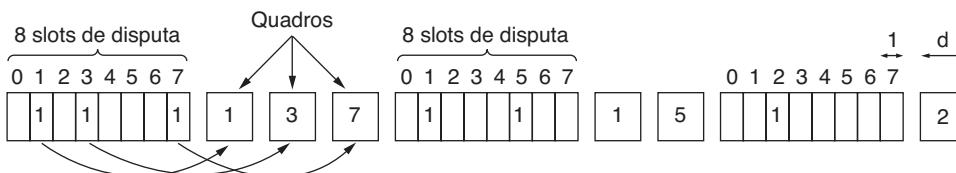



Figura 4.6 | O protocolo básico bit-map.

transmiti-lo em outra, como vemos na Figura 4.7. Os quadros também são transmitidos na direção do token. Desse modo, eles circularão em torno do anel e alcançarão qualquer estação que seja o destino. Contudo, para impedir que o quadro circule indefinidamente (assim como o próprio token), alguma estação precisa removê-lo do anel. Essa estação pode ser a que enviou o quadro originalmente, depois que ele passou por um ciclo completo, ou a estação de destino do quadro.

Observe que não precisamos de um anel físico para implementar a passagem de tokens. Em vez disso, o canal que conecta as estações poderia ser um único e longo barramento. Em seguida, cada estação usa o barramento para enviar o token para a próxima estação em uma sequência predefinida. A posse do token permite que uma estação use o barramento para enviar um quadro, como antes. Esse protocolo é chamado **barramento de tokens** (ou **token bus**).

O desempenho da passagem de tokens é semelhante ao do protocolo bit-map, embora os slots de disputa e os quadros de um ciclo agora estejam embaralhados. Depois de enviar um quadro, cada estação precisa esperar que todas as N estações (incluindo ela mesma) transmitam o token aos seus vizinhos e as outras $N - 1$ estações transmitam um quadro, se tiverem um. Uma diferença sutil é que, como todas as posições no ciclo são equivalentes, não existe parcialidade para estações com numeração baixa ou alta. Para o token ring, cada estação também está apenas transmitindo o token, enquanto sua estação vizinha anterior no protocolo realiza o passo seguinte. Cada token não precisa se propagar para todas as estações antes que o protocolo avance para o passo seguinte.

Os token rings surgiram como protocolos MAC com certa consistência. Um antigo protocolo token ring (chamado "Token Ring" e padronizado como IEEE 802.5) era popular na década de 80 como uma alternativa à Ethernet clássica. Na década de 90, um token ring muito mais rápido, chamado **FDDI** (**Fiber Distributed Data Interface**), foi extinto pela Ethernet comutada. Na década de 2000, um token ring chamado **RPR** (**Resilient Packet Ring**) foi definido como IEEE 802.17 para padronizar a mistura de anéis metropolitanos em uso pelos ISPs. Ainda veremos o que a próxima década nos oferecerá.

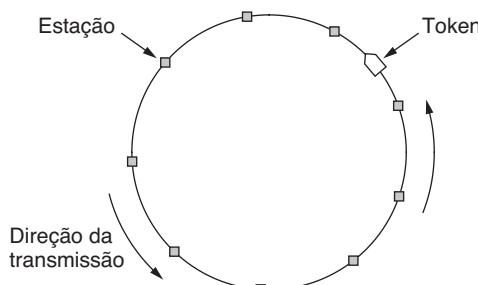



Figura 4.7 | Token ring.

Contagem regressiva binária

Um problema com o protocolo básico bit-map, e também com a passagem de tokens, é que o overhead é de 1 bit por estação e, portanto, ele não se adapta muito bem a redes com milhares de estações. Podemos fazer melhor que isso usando endereços binários de estações com um canal que combine transmissões. Uma estação que queira usar o canal transmite seu endereço como uma sequência de bits binários, começando com o bit de alta ordem. Supomos que todos os endereços têm o mesmo tamanho. Os bits de cada posição de endereço das diferentes estações passam juntos por uma operação OR booleana pelo canal quando são enviados ao mesmo tempo. Chamaremos esse protocolo de **contagem regressiva binária**. Ele foi usado no Datakit (Fraser, 1987). Esse protocolo pressupõe implicitamente que os atrasos de transmissão são desprezíveis, de forma que todas as estações detectam bits declarados quase instantaneamente.

Para evitar conflitos, é necessário que seja aplicada uma regra de arbitragem: assim que uma estação percebe que um bit de alta ordem que em seu endereço era 0 foi sobrescrito por um bit 1, ela desiste. Por exemplo, se as estações 0010, 0100, 1001 e 1010 estiverem todas tentando acessar o canal, no primeiro período de um bit, as estações transmitirão 0, 0, 1 e 1, respectivamente. Esses valores passarão pela operação OR para formar um valor 1. As estações 0010 e 0100 veem o valor 1 e sabem que uma estação de numeração mais alta está disputando o canal e, portanto, desistem da luta na rodada atual. As estações 1001 e 1010 prosseguem.

O próximo bit é 0, e ambas as estações continuam a transmissão. O próximo bit é 1 e, portanto, a estação 1001 desiste. A vencedora é a estação 1010, pois tem o endereço mais alto. Após vencer a disputa, é provável que agora ela possa transmitir um quadro, após o qual terá início outro ciclo de disputa. A Figura 4.8 ilustra esse protocolo. Ele tem a propriedade de dar às estações com numeração mais alta uma prioridade maior do que a prioridade concedida a estações de numeração mais baixa, o que pode ser bom ou ruim, dependendo do contexto.

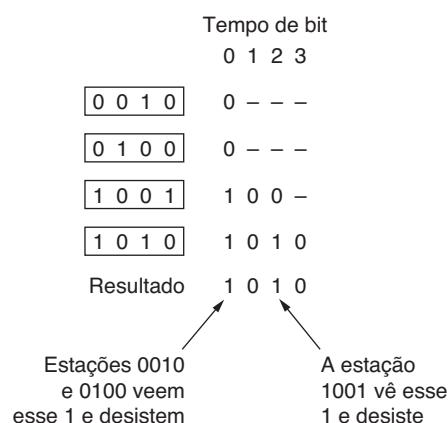



Figura 4.8 | O protocolo de contagem regressiva binária. Um traço significa inatividade.

Com esse método, a eficiência do canal é $d/(d + \log_2 N)$. No entanto, se o formato do quadro tiver sido corretamente escolhido, de forma que o endereço do transmissor seja o primeiro campo do quadro, mesmo esses $\log_2 N$ bits não serão desperdiçados, e a eficiência será de 100 por cento.

A contagem regressiva binária é um exemplo de protocolo simples, elegante e eficiente, que está esperando o momento de ser redescoberto. Esperamos que algum dia ele encontre um novo lar.

4.2.4 Protocolos de disputa limitada

Já vimos até agora duas estratégias básicas para a aquisição de canais em uma rede de broadcast: métodos de disputa, como no CSMA, e métodos livres de colisão. Cada estratégia é classificada de acordo com seu desempenho em relação a duas medidas importantes: o atraso em carga baixa e a eficiência de canal em carga alta. Em condições de carga leve, a disputa (ou seja, o ALOHA original ou o slotted ALOHA) é preferível, em virtude de seu baixo índice de atraso (pois as colisões são raras). À medida que a carga aumenta, a disputa torna-se cada vez menos interessante, pois o overhead associado à arbitragem do canal torna-se maior. O oposto também é verdadeiro em relação aos protocolos livres de colisão. Em carga baixa, eles têm um alto índice de atraso, mas, à medida que a carga aumenta, a eficiência do canal melhora (pois os overheads são fixos).

Obviamente, seria bom se pudéssemos combinar as melhores propriedades dos protocolos de disputa e dos protocolos livres de colisão, chegando a um novo protocolo que usaria não só a disputa em cargas baixas, para proporcionar um baixo índice de atraso, mas também a técnica livre de colisão em carga alta, para oferecer uma boa eficiência de canal. Esses protocolos, que chamaremos **protocolos de disputa limitada**, de fato existem, e concluirão nosso estudo sobre redes com detecção de portadora.

Até agora, os únicos protocolos de disputa que estudamos são simétricos. Ou seja, cada estação tenta acessar o canal com a mesma probabilidade, p, com todas as estações usando o mesmo p. É interessante observar que o desempenho geral do sistema às vezes pode ser melhorado com o uso de um protocolo que atribua probabilidades distintas a diferentes estações.

Antes de examinarmos os protocolos assimétricos, faremos uma pequena revisão do desempenho do caso simétrico. Suponha que k estações estejam disputando o acesso a um canal. Cada uma tem a probabilidade p de transmitir durante cada slot. A probabilidade de alguma estação acessar o canal com sucesso durante determinado slot é a probabilidade de que qualquer estação transmita, com probabilidade p, e todas as outras $k - 1$ estações adiem, cada uma com probabilidade $1 - p$. Esse valor é $kp(1 - p)^{k-1}$. Para encontrar o valor ideal de p, diferenciamos em relação a p, definimos o resultado como zero e resolvemos a equação para p. Ao fazer isso, descobrimos que o melhor valor de p é $1/k$. Ao substituirmos $p = 1/k$, obteremos:

$$\Pr[\text{sucesso com } p \text{ ideal}] = \left[\frac{k-1}{k}\right]^{k-1} \tag{4.4}$$

Essa probabilidade está representada na Figura 4.9. Para um pequeno número de estações, as chances de sucesso são boas, mas, tão logo o número de estações alcance até mesmo cinco, a probabilidade cai até um número próximo de seu valor assintótico, $1/e$.

Pela Figura 4.9, fica evidente que a probabilidade de alguma estação adquirir o canal só pode ser aumentada diminuindo-se o volume de competição. Os protocolos de disputa limitada fazem exatamente isso. Primeiro, eles dividem as estações em grupos (não necessariamente disjuntos). Apenas os membros do grupo 0 podem disputar o slot 0. Se um deles obtiver êxito, adquirirá o canal e transmitirá seu quadro. Se um slot permanecer inativo ou se ocorrer uma colisão, os membros do grupo 1 disputarão o slot 1 etc. Fazendo-se uma divisão apropriada das estações em grupos, o volume de disputa por cada slot pode ser reduzido e, assim, a operação de cada slot ficará próxima à extremidade esquerda da Figura 4.9.

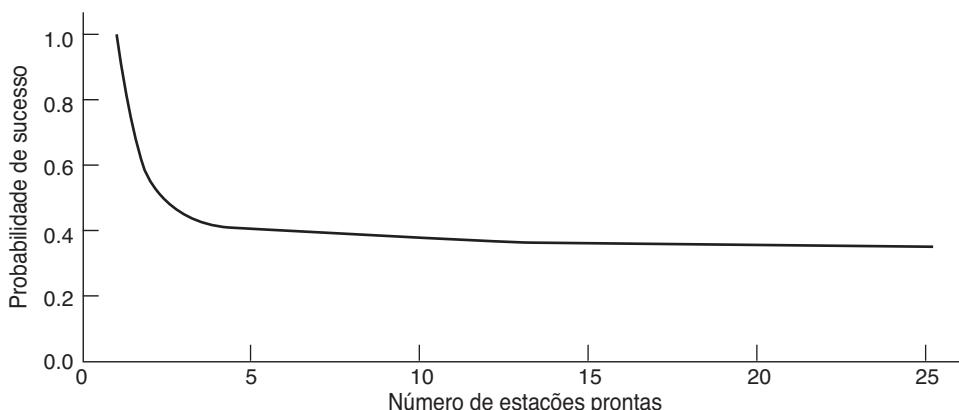



Figura 4.9 | Probabilidade de aquisição de um canal de disputa simétrico.

O truque é a maneira de atribuir estações a slots. Antes de analisarmos o caso geral, vamos considerar algumas situações especiais. Em um extremo, cada grupo tem apenas um membro. Essa atribuição garante que nunca ocorrerão colisões, pois existirá, no máximo, uma estação disputando qualquer slot dado. Já vimos esse tipo de protocolo antes (por exemplo, a contagem regressiva binária). A próxima situação especial é atribuir duas estações por grupo. A probabilidade de ambas tentarem transmitir durante um slot é p^2, que, para um p pequeno, é desprezível. À medida que mais e mais estações são atribuídas ao mesmo slot, a probabilidade de colisão aumenta, mas diminui a extensão da varredura de bit-map necessária para que todas tenham uma chance. A situação-limite consiste em um único grupo que contém todas as estações (ou seja, o slotted ALOHA). O que precisamos é de uma forma de atribuir dinamicamente estações a slots, com várias estações por slot quando a carga for baixa, e poucas estações (ou apenas uma) por slot quando a carga for alta.

O protocolo adaptativo tree walk

Uma maneira particularmente simples de fazer as atribuições necessárias consiste em usar o algoritmo desenvolvido pelo exército norte-americano para testar a incidência de sífilis em soldados durante a Segunda Guerra Mundial (Dorfman, 1943). Em resumo, o exército extraiu uma amostra de sangue de N soldados. Uma parte de cada amostra foi colocada em um único tubo de teste. Essa amostra misturada foi submetida a teste par detectar anticorpos. Se nenhum anticorpo fosse encontrado, todos os soldados do grupo eram considerados saudáveis. Se houvesse anticorpos, duas novas amostras misturadas eram preparadas, uma dos soldados numerados de 1 a $N/2$ e outra com o sangue dos demais soldados. O processo era repetido recursivamente até que os soldados infectados fossem identificados.

Para a versão computacional desse algoritmo (Capetanakis, 1979), é conveniente imaginar as estações como as folhas de uma árvore binária, conforme ilustra a Figura 4.10. No primeiro slot de disputa que segue uma transmissão de quadro bem-sucedida, o slot 0, todas as estações têm permissão para tentar acessar o canal. Se uma delas conseguir, muito bem. Se ocorrer uma colisão, durante o slot 1, apenas as estações que estiverem sob o nó 2 da árvore poderão disputar o canal. Se uma delas se apoderar do canal, o slot seguinte ao quadro ficará reservado para as estações do nó 3. Por outro lado, se duas ou mais estações no nó 2 quiserem transmitir, ocorrerá uma colisão durante o slot 1 e, nesse caso, será a vez do nó 4 durante o slot 2.

Basicamente, se ocorrer uma colisão durante o slot 0, toda a árvore será pesquisada, primeiro em profundidade, a fim de localizar todas as estações prontas para transmissão. Cada slot de bits é associado a algum nó específico da árvore. Se ocorrer uma colisão, a pesquisa continuará recursivamente com os filhos localizados à esquerda e à direita desse nó. Se um slot de bits estiver inativo ou se houver apenas uma estação transmitindo nesse slot, a pesquisa de seu nó poderá ser encerrada, pois todas as estações prontas terão sido localizadas. (Se houvesse mais de uma, uma colisão teria ocorrido.)

Quando a carga do sistema está muito pesada, quase não vale a pena o esforço de dedicar o slot 0 ao nó 1, pois esse procedimento só faz sentido na eventualidade improvável de que exatamente uma estação tenha um quadro a ser transmitido. Assim, alguém poderia argumentar que os nós 2 e 3 também deveriam ser ignorados, pela mesma razão. Em termos mais gerais, em que nível da árvore a pesquisa deve ter início? É claro que, quanto maior for a carga, mais baixo na árvore o ponto de início da pesquisa deve estar. Por ora, vamos supor que cada estação tenha uma boa estimativa do número q de estações prontas, por exemplo, com base no monitoramento de tráfego mais recente.

Para prosseguir, vamos numerar os níveis da árvore a partir do topo, com o nó 1 da Figura 4.10 no nível 0, os nós 2 e 3 no nível 1 etc. Observe que cada nó do nível i tem uma fração 2^{-i} das estações que se encontram abaixo dele. Se as q estações prontas estiverem uniformemente distribuídas, o número esperado dessas estações abaixo de um nó específico do nível i será apenas $2^{-i} q$. Intuitivamente, seria de esperar que o nível ideal para iniciar a

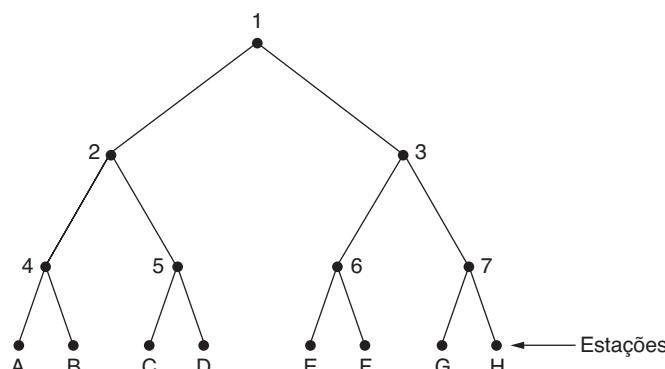



Figura 4.10 | A árvore para oito estações.

pesquisa na árvore fosse aquele no qual o número médio de estações em disputa por slot fosse igual a 1, isto é, o nível em que $2^{-i} q = 1$. Resolvendo essa equação, descobrimos que $i = \log_2 q$.

Foram descobertas diversas melhorias no algoritmo básico, as quais são abordadas em detalhes por Bertsekas e Gallager (1992). Por exemplo, considere a hipótese em que as estações *G* e *H* são as únicas que estão esperando para transmitir. No nó 1, ocorrerá uma colisão, de modo que 2 será testado e descoberto como nó inativo. É inútil testar o nó 3, pois é certo que haverá colisão (sabemos que duas ou mais estações abaixo de 1 estão prontas e que nenhuma delas está abaixo de 2; portanto, todas devem estar abaixo de 3). A sondagem do nó 3 pode ser ignorada, e o nó 6 será testado em seguida. Quando essa sondagem também não produzir nenhum resultado, 7 poderá ser ignorado e o nó *G* poderá ser testado em seguida.

4.2.5 Protocolos de LANs sem fios

Um sistema de notebooks que se comunicam por rádio pode ser considerado uma LAN sem fios, como discutimos na Seção 1.5.3. Essa LAN é um exemplo de canal de broadcast. Ela também tem propriedades um pouco diferentes das que caracterizam as LANs com fios, o que leva a diferentes protocolos MAC. Nesta seção, analisaremos alguns desses protocolos. Na Seção 4.4, examinaremos a rede 802.11 (WiFi) em detalhes.

Uma configuração comum para uma LAN sem fios é um edifício comercial com pontos de acesso (PAs) estrategicamente posicionados. Todos os PAs são interconectados com o uso de cobre ou fibra, para melhorar a conectividade com as estações que falam com eles. Se a potência de transmissão dos PAs e dos notebooks for ajustada para um alcance de dezenas de metros, as salas vizinhas se tornarão uma única célula e o edifício inteiro passará a ser um grande sistema celular, como os que estudamos no Capítulo 2, exceto que cada célula só tem um canal, que cobre toda a largura de banda disponível e todas as estações em sua célula, incluindo o PA. Em geral, ela oferece larguras de banda de 1 megabit/s até 600 Mbps.

Já notamos que os sistemas sem fios normalmente não podem detectar uma colisão enquanto ela está ocorrendo. O sinal recebido em uma estação pode ser curto, talvez um milhão de vezes mais fraco que o sinal que está sendo transmitido. Encontrá-lo é como procurar uma onda no oceano. Em vez disso, as confirmações são usadas para descobrir colisões e outros erros após o fato.

Há uma diferença ainda mais importante entre as LANs sem fios e as convencionais. Uma estação em uma LAN sem fios pode não ser capaz de transmitir quadros ou recebê-los de todas as estações, em decorrência do alcance de rádio limitado das estações. Nas LANs com fios, quando uma estação envia um quadro, todas as outras estações o recebem. A ausência dessa propriedade nas LANs sem fios causa uma série de complicações.

Vamos simplificar supondo que cada transmissor de rádio tenha algum alcance fixo, representado por uma região de cobertura circular dentro da qual outra estação possa detectar e receber a transmissão da estação. É importante observar que, na prática, as regiões de cobertura não são tão regulares, pois a propagação dos sinais de rádio depende do ambiente. As paredes e outros obstáculos que atenuam e refletem sinais podem fazer com que o alcance seja bastante diferente em diferentes direções. Mas um modelo circular simples servirá aos nossos propósitos.

Uma abordagem simples para o uso de uma LAN sem fios seria tentar o CSMA: basta escutar outras transmissões e transmitir apenas se ninguém mais estiver usando o canal. O problema é que esse protocolo realmente não é uma boa maneira de pensar nas redes sem fios, pois o que importa para a recepção é a interferência no receptor, e não no transmissor. Para ver a natureza do problema, considere a Figura 4.11, na qual ilustramos quatro estações sem fios. Para os nossos propósitos, não importa quais são PAs e quais são notebooks. O alcance do rádio é tal que *A* e *B* estão dentro do alcance um do outro e podem interferir um no outro. *C* também pode interferir em *B* e *D*, mas não em *A*.

Considere primeiro o que acontece quando *A* está transmitindo para *B*, como mostra a Figura 4.11(a). Se *A* transmitir e depois *C* imediatamente detectar o meio físico, ele não ouvirá *A*, pois essa estação está fora do alcance e, portanto, concluirá incorretamente que pode transmitir para *B*. Se começar a transmitir, *C* interferirá em *B*, removendo o quadro de *A*. (Consideramos aqui que nenhum esquema tipo CDMA é usado para oferecer múltiplos canais, de modo que as colisões inutilizam o sinal e destroem os dois quadros.) Queremos um protocolo MAC que impeça esse tipo de colisão, pois isso desperdiça largura de banda.

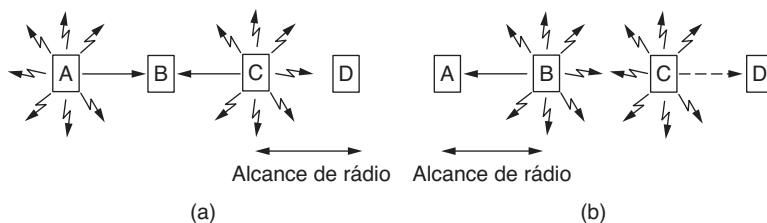



Figura 4.11 | Uma LAN sem fios. (a) *A* e *C* são terminais ocultos ao transmitir para *B*. (b) *B* e *C* são terminais expostos ao transmitir para *A* e *D*.

O problema de uma estação não conseguir detectar uma provável concorrente pelo meio físico, porque a estação concorrente está muito longe, é denominado **problema da estação oculta**.

Agora, vamos considerar a situação inversa: *B* está transmitindo para *A* ao mesmo tempo que *C* deseja transmitir para *D*, como mostra a Figura 4.11(b). Se detectar o meio físico, *C* ouvirá uma transmissão em andamento e concluirá incorretamente que não pode transmitir para *D* (como mostra a linha tracejada). Na verdade, essa transmissão só geraria uma recepção de má qualidade na zona entre *B* e *C*, em que nenhum dos receptores desejados está localizado. Queremos um protocolo MAC que impeça esse tipo de adiamento, pois ele desperdiça largura de banda. Essa situação é chamada **problema da estação exposta**.

O problema é que, antes de iniciar uma transmissão, a estação realmente deseja saber se há ou não atividade de rádio em torno do receptor. O CSMA apenas informa a ela se há ou não atividade na estação que detecta a portadora. Com um fio, todos os sinais se propagam para todas as estações e, portanto, não existe distinção. Porém, somente uma transmissão pode ocorrer de cada vez em qualquer parte do sistema. Em um sistema baseado em ondas de rádio de pequeno alcance, várias transmissões podem ocorrer simultaneamente, se todas tiverem destinos diferentes e esses destinos estiverem fora do alcance uns dos outros. Queremos que essa concorrência aconteça quando a célula se tornar cada vez maior, da mesma forma que pessoas em uma festa não devem esperar que todos na sala fiquem em silêncio antes de começar a falar; várias conversas podem ocorrer ao mesmo tempo em uma sala grande, desde que elas não sejam dirigidas para o mesmo local.

Um protocolo antigo e influente, que trata desses problemas em LANs sem fios, é o **acesso múltiplo com prevenção de colisão**, ou **MACA** (**Multiple Access with Collision Avoidance**) (Karn, 90). A ideia básica consiste em fazer com que o transmissor estimule o receptor a liberar um quadro curto como saída, de modo que as estações vizinhas possam detectar essa transmissão e evitar transmitir enquanto o quadro de dados (grande) estiver sendo recebido. Essa técnica é usada no lugar da detecção de portadora.

O MACA é ilustrado na Figura 4.12. Vamos analisar agora como *A* envia um quadro para *B*. *A* inicia a transmissão enviando um quadro de **solicitação para envio**, ou **RTS** (**Request to Send**), para *B*, como mostra a Figura 4.12(a). Esse quadro curto (30 bytes) contém o comprimento do quadro de dados que possivelmente será enviado em seguida. Depois disso, *B* responde com um quadro de **liberação para envio**, ou **CTS** (**Clear to Send**), como mostra a Figura 4.12(b). O quadro CTS contém o tamanho dos dados (copiado do quadro RTS). Após o recebimento do quadro CTS, *A* inicia a transmissão.

Agora, vejamos como reagem as estações que estão ouvindo esses quadros por acaso. Qualquer estação que esteja ouvindo o quadro RTS está próxima de *A* e deve permanecer inativa por tempo suficiente para que o CTS seja transmitido de volta para *A*, sem conflito. Qualquer estação que esteja ouvindo o CTS está próxima de *B* e deve permanecer inativa durante a transmissão de dados que está a caminho, cujo tamanho pode ser verificado pelo exame do quadro CTS.

Na Figura 4.12, *C* está dentro do alcance de *A*, mas não no alcance de *B*. Portanto, essa estação pode detectar a RTS de *A*, mas não a CTS de *B*. Desde que não interfira na CTS, a estação é livre para transmitir enquanto o quadro de dados está sendo enviado. Ao contrário, *D* está dentro do alcance de *B*, mas não de *A*. Ela não detecta a RTS, mas sim a CTS. Ao detectá-la, ela recebe a indicação de que está perto de uma estação que está prestes a receber um quadro e, portanto, adia a transmissão até o momento em que a transmissão desse quadro provavelmente esteja concluída. A estação *E* detecta as duas mensagens de controle e, como *D*, deve permanecer inativa até que a transmissão do quadro de dados seja concluída.

Apesar dessas precauções, ainda pode haver colisões. Por exemplo, *B* e *C* poderiam enviar quadros RTS para *A* ao mesmo tempo. Haverá uma colisão entre esses quadros

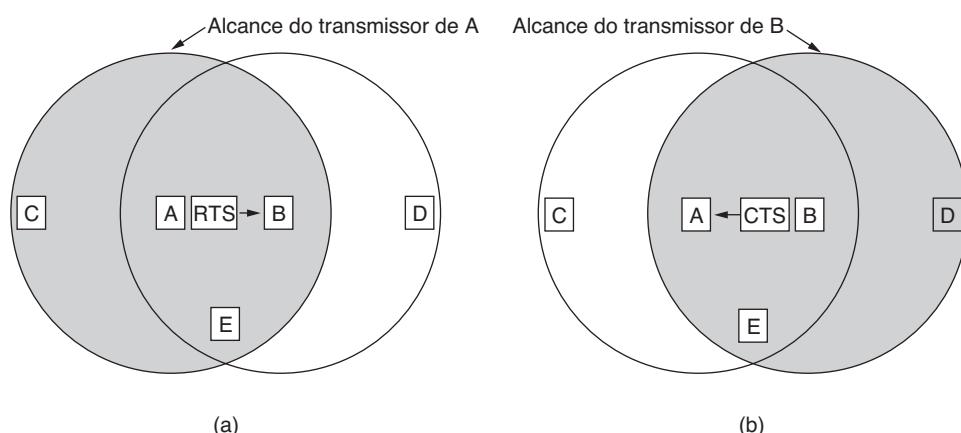



Figura 4.12 | O protocolo MACA. (a) *A* está enviando um quadro RTS para *B*. (b) *B* está respondendo com um quadro CTS para *A*.

e eles se perderão. No caso de uma colisão, um transmissor que não obtiver êxito (ou seja, o que não detectar uma CTS no intervalo esperado) aguardará durante um intervalo aleatório e tentará novamente mais tarde.

4.3 Ethernet

Agora, concluímos nossa discussão resumida sobre protocolos de alocação de canais e, portanto, é hora de analisar como esses princípios se aplicam a sistemas reais. Muitos dos projetos para redes pessoais, locais e metropolitanas foram padronizados com o nome IEEE 802. Alguns desses padrões sobreviveram, mas muitos não, como vimos na Figura 1.38. Algumas pessoas que acreditam em reencarnação creem que Charles Darwin retornou como membro da associação de padrões do IEEE com a finalidade de eliminar os menos capazes. Os mais importantes entre os sobreviventes são os padrões 802.3 (Ethernet) e 802.11 (LAN sem fios). O Bluetooth (PAN sem fios) é bastante utilizado, mas agora foi padronizado fora do 802.15. Com o 802.16 (MAN sem fios), ainda é cedo para dizer. Consulte a próxima edição deste livro para descobrir.

Começaremos nosso estudo dos sistemas reais com a Ethernet, provavelmente o tipo de rede de computação mais utilizado no mundo. Existem dois tipos de Ethernet: **Ethernet clássica**, que resolve o problema de acesso múltiplo por meio de técnicas que estudamos neste capítulo; e **Ethernet comutada**, em que dispositivos chamados switches são usados para conectar diferentes computadores. É importante observar que, embora ambas sejam chamadas Ethernet, elas são muito diferentes. A Ethernet clássica é a forma original, que atuava em velocidades de 3 a 10 Mbps. A Ethernet comutada é a evolução da Ethernet, e trabalha em velocidades de 100, 1.000 e 10.000 Mbps, ao que chamamos Fast Ethernet, gigabit Ethernet e 10 gigabit Ethernet. Na prática, somente a Ethernet comutada é usada atualmente.

Discutiremos essas formas históricas da Ethernet em ordem cronológica, mostrando como elas se desenvolveram. Como Ethernet e IEEE 802.3 são idênticos, exceto por uma pequena diferença (que discutiremos em breve), muitas pessoas usam os termos 'Ethernet' e 'IEEE 802.3' para indicar a mesma coisa. Também faremos isso aqui. Para obter mais informações sobre Ethernet, consulte Spurgeon (2000).

4.3.1 Camada física da Ethernet clássica

A história da Ethernet começa mais ou menos na época da ALOHA, quando um aluno chamado Bob Metcalfe conseguiu seu título de bacharel no MIT e depois 'subiu o rio' para obter seu título de Ph.D. em Harvard. Durante seus estudos, ele conheceu o trabalho de Abramson. Ele ficou tão interessado que, depois de se formar em Harvard, decidiu passar o verão no Havaí trabalhando com Abramson, antes de iniciar seu trabalho no Xerox PARC (Palo Alto Research Center). Quando chegou ao PARC, viu que os pesquisadores de lá haviam projetado e montado o que mais tarde seriam chamados computadores pessoais. Mas as máquinas eram isoladas. Usando seu conhecimento do trabalho de Abramson, Metcalfe, com seu colega David Boggs, projetou e implementou a primeira rede local (Metcalfe e Boggs, 1976). Ele usou um único cabo coaxial grosso e conseguiu trabalhar a 3 Mbps.

Metcaffe e Boggs chamaram o sistema de **Ethernet**, fazendo referência ao *éter transmissor de luz* (do inglês *luminiferous ether*), através do qual se acreditava que a radiação eletromagnética se propagava. (Quando o físico britânico do século XIX James Clerk Maxwell descobriu que a radiação eletromagnética poderia ser descrita por uma equação de onda, os cientistas acharam que o espaço deveria estar repleto de algum meio etéreo em que a radiação estava se propagando. Somente depois do famoso experimento de Michelson-Morley, em 1887, é que os físicos descobriram que a radiação eletromagnética podia se propagar no vácuo.)

A rede Ethernet da Xerox foi tão bem-sucedida que DEC, Intel e Xerox chegaram a um padrão em 1978 para uma Ethernet de 10 Mbps, chamado **padrão DIX**. Com uma pequena mudança, o padrão DIX tornou-se o padrão IEEE 802.3 em 1983. Infelizmente para a Xerox, ela já tinha um histórico de criar invenções originais (como o computador pessoal) e não conseguir comercializá-las, história contada em *Fumbling the Future* [Tateando o futuro], de Smith e Alexander (1988). Quando a Xerox mostrou pouco interesse em fazer algo com a Ethernet além de ajudar a padronizá-la, Metcalfe formou sua própria empresa, a 3Com, para vender adaptadores Ethernet para PCs. Ele vendeu muitos milhões deles.

A Ethernet clássica percorria o prédio como um cabo longo único, ao qual todos os computadores eram conectados. Essa arquitetura é mostrada na Figura 4.13. A primeira variedade, popularmente conhecida como **thick Ethernet**, era semelhante a uma mangueira amarela de jardim, com marcações a cada 2,5 metros, mostrando onde conectar os computadores. (O padrão 802.3 não *exigia* realmente que o cabo fosse amarelo, mas *sugeria* isso.) Ela foi acompanhada pela **thin Ethernet**, que encurvava com mais facilidade e fazia conexões usando conectores BNC padrão da indústria. A thin Ethernet era muito mais barata e fácil de instalar, mas só podia ter 185 metros por segmento (em vez dos 500 m da thick Ethernet), cada um dos quais podendo lidar com apenas trinta máquinas (em vez de cem).

Cada versão da Ethernet tem um comprimento máximo de cabo por segmento (ou seja, comprimento não amplificado) sobre o qual o sinal será propagado. Para permitir redes maiores, vários cabos podem ser conectados por **repetidores**. Um repetidor é um dispositivo da camada física que recebe, amplifica (ou seja, regenera) e retransmite sinais nas duas direções. Em relação ao software, diversos segmentos de cabo conectados por repetidores não são di-

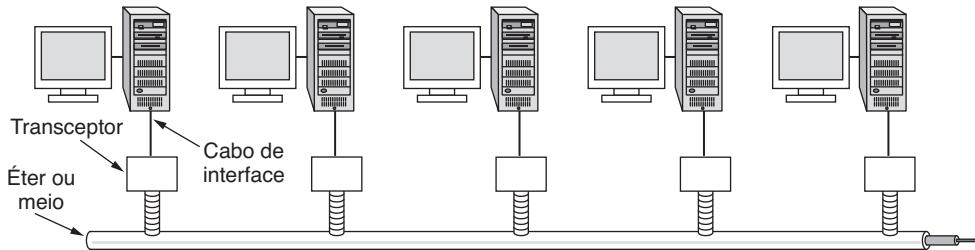



Figura 4.13 | Arquitetura da Ethernet clássica.

ferentes de um único cabo (exceto por um pequeno atraso introduzido pelos repetidores).

Por um a um desses cabos, a informação era enviada usando a codificação Manchester que estudamos na Seção 2.5. Uma Ethernet poderia conter vários segmentos de cabo e vários repetidores, mas dois transceptores não poderiam estar mais de 2,5 km afastados um do outro e nenhum caminho entre dois transceptores quaisquer poderia atravessar mais de quatro repetidores. O motivo para essa restrição foi para que o protocolo MAC, que examinaremos em seguida, funcionasse corretamente.

4.3.2 O protocolo da subcamada MAC Ethernet

O formato usado para transmitir quadros é mostrado na Figura 4.14. Cada quadro começa com um *Preâmbulo* de 8 bytes, cada um contendo o padrão de bits 10101010 (com exceção do último byte, em que os dois últimos bits são 11). Esse último byte é chamado de delimitador de *Início de quadro* para o 802.3. A codificação Manchester desse padrão produz uma onda quadrada de 10 MHz por 6,4 μs, a fim de permitir a sincronização entre o clock do receptor e o clock do transmissor. Os dois últimos bits 1 dizem ao receptor que o restante do quadro está para começar.

Em seguida, o quadro contém dois endereços, um para o destino e um para a origem. Cada um deles possui 6 bytes de extensão. O primeiro bit transmitido do endereço de destino é 0 para endereços comuns e 1 para endereços de grupos. Estes permitem que diversas estações escutem um único endereço. Quando um quadro é enviado para um endereço de grupo, todas as estações do grupo o recebem. A transmissão para um grupo de estações é chamada **multicasting**. O endereço que consiste em todos os bits 1 é reservado para **broadcasting**. Um quadro contendo todos os bits 1 no campo de destino é aceito por todas as estações da rede. O multicasting é mais seletivo, mas envolve o gerenciamento de grupos para definir quais estações pertencem ao grupo. Por outro lado, o broadcasting não diferencia entre estação alguma e, por isso, não requer nenhum gerenciamento de grupos.

Uma característica interessante dos endereços de origem da estação é que eles são globalmente exclusivos, atribuídos de forma centralizada pelo IEEE para garantir que duas estações em qualquer lugar do mundo nunca tenham o mesmo endereço. A ideia é que qualquer estação possa endereçar de forma exclusiva qualquer outra estação simplesmente informando o número de 48 bits correto. Para fazer isso, os três primeiros bytes do campo de endereço são usados para um **identificador exclusivo da organização**, ou **OUI** (**Organizationally Unique Identifier**). Os valores para esse campo são atribuídos diretamente pelo IEEE e indicam o fabricante. Os fabricantes recebem blocos de 2^{24} endereços. O fabricante atribui os três últimos bytes do endereço e programa o endereço completo na NIC antes que ela seja vendida.

Em seguida, vem o campo *Tipo* ou *Tamanho*, dependendo se o quadro é Ethernet ou IEEE 802.3. A Ethernet usa um campo *Tipo* para informar ao receptor o que fazer com o quadro. Vários protocolos da camada de rede podem estar em uso ao mesmo tempo na mesma máquina; assim, quando chega um quadro Ethernet, o sistema operacional tem de saber a qual deles deve entregar o quadro. O campo *Tipo* especifica que processo deve receber o quadro. Por exemplo, um código tipo 0x0800 significa que os dados contêm um pacote IPv4.

O IEEE 802.3, em sua sabedoria, decidiu que esse campo transportaria o tamanho do quadro, pois o tama-

Bytes	8		6	6	2	0-1500	0-46	4
(a)	Preâmbulo		Endereço de destino	Endereço de origem	Tipo	Dados	Preenchimento	Check-sum
(b)	Preâmbulo	I d q	Endereço de destino	Endereço de origem	Tamanho	Dados	Preenchimento	Check-sum

Figura 4.14 | Formato dos quadros. (a) Ethernet (DIX). (b) IEEE 802.3.

nho Ethernet era determinado examinando o interior dos dados — uma violação do uso de camadas, se é que isso existiu. Naturalmente, isso significava que não havia como o receptor descobrir o que fazer com um quadro que chegava. Esse problema foi tratado pelo acréscimo de outro cabeçalho para o protocolo de **controle lógico do enlace**, ou **LLC** (**Logical Link Control**), dentro dos dados. Ele usa 8 bytes para transportar os 2 bytes de informação do tipo de protocolo.

Infelizmente, quando o 802.3 foi publicado, já havia tanto hardware e software para a Ethernet DIX em uso que poucos fabricantes e usuários tiveram interesse em modificar os campos de *Tipo* e *Tamanho*. Em 1997, o IEEE jogou a toalha e disse que as duas maneiras poderiam ser usadas. Felizmente, todos os campos de *Tipo* em uso antes de 1997 tinham valores maiores que 1500, que ficou bem estabelecido como o tamanho máximo dos dados. Agora, a regra é que qualquer número menor ou igual a 0x600 (1536) pode ser interpretado como *Tamanho*, ao passo que qualquer número maior que 0x600 pode ser interpretado como *Tipo*. Agora o IEEE pode afirmar que todos estão usando seu padrão e todos os outros podem continuar fazendo o que já faziam (não se preocupar com o LLC) sem se sentir culpados por isso.

Depois, vêm os dados, com até 1.500 bytes. Esse limite foi escolhido de forma um tanto arbitrária na época em que o padrão Ethernet foi esculpido, principalmente com base no fato de que um transceptor precisa ter RAM suficiente para guardar um quadro inteiro e, em 1978, a RAM tinha um custo muito alto. Um limite superior maior significaria mais RAM e, consequentemente, um transceptor mais caro.

Além de haver um comprimento máximo de quadro, também existe um comprimento mínimo. Embora um campo de dados de 0 bytes às vezes seja útil, ele causa um problema. Quando um transceptor detecta uma colisão, ele trunca o quadro atual, o que significa que bits perdidos e fragmentos de quadros aparecem a todo instante no cabo. Para tornar mais fácil a distinção entre quadros válidos e lixo, o padrão Ethernet exige que os quadros válidos tenham pelo menos 64 bytes de extensão, do endereço de destino até o campo de checksum, incluindo ambos. Se a parte de dados de um quadro for menor que 46 bytes, o campo *Preenchimento* será usado para preencher o quadro até o tamanho mínimo.

Outra (e mais importante) razão para a existência de um quadro de comprimento mínimo é impedir que uma estação conclua a transmissão de um quadro curto antes de o primeiro bit ter atingido a outra extremidade do cabo, em que ele poderá colidir com outro quadro. Esse problema é ilustrado na Figura 4.15. No tempo 0, a estação *A*, localizada em uma extremidade da rede, envia um quadro. Vamos chamar de τ o tempo de propagação que esse quadro leva para atingir a outra extremidade. Momentos antes de o quadro chegar à outra extremidade (ou seja, no tempo $\tau - \varepsilon$), a estação mais distante, *B*, inicia a transmissão. Quando detecta que está recebendo mais potência do que está transmitindo, *B* sabe que ocorreu uma colisão, interrompe a transmissão e gera uma rajada de sinal ruidoso de 48 bits para avisar a todas as outras estações. Em outras palavras, ela bloqueia o éter (meio) para ter certeza de que o transmissor não ignorará a colisão. Aproximadamente no tempo 2τ, o transmissor detecta a rajada de ruídos e também cancela sua transmissão. Depois, ele espera por um tempo aleatório antes de tentar novamente.

Se uma estação tentar transmitir um quadro muito curto, é concebível que ocorra uma colisão, mas a transmissão será concluída antes que a rajada de sinal ruidoso retorne no instante 2τ. Então, o transmissor concluirá incorretamente que o quadro foi enviado com êxito. Para evitar que essa situação ocorra, a transmissão de todos os quadros deve demorar mais de 2τ para transmitir, de forma que a transmissão ainda esteja acontecendo quando a rajada de sinal ruidoso voltar ao transmissor. Para uma LAN de 10 Mbps com um comprimento máximo de 2.500 metros e quatro repetidores (de acordo com a especificação 802.3), o tempo de ida e volta (incluindo o tempo de propagação pelos quatro repetidores) foi calculado em quase 50 μs na pior das hipóteses. Portanto, o quadro mínimo deve demorar pelo menos esse tempo para ser transmitido. A 10 Mbps,

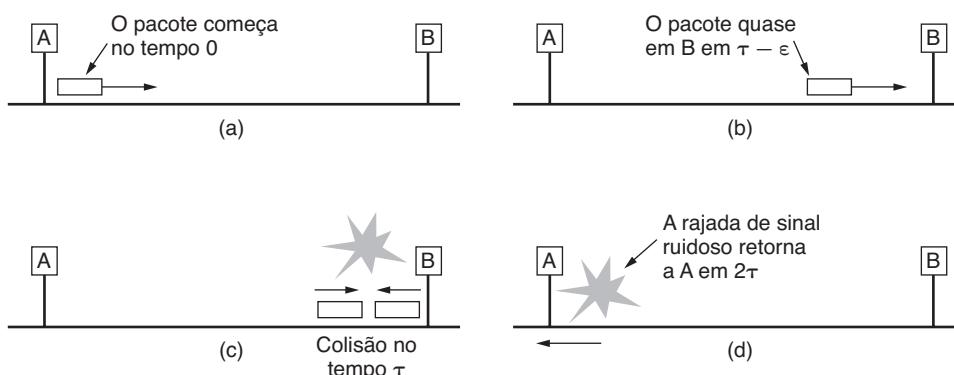



Figura 4.15 | A detecção de colisão pode demorar até o tempo 2τ.

um bit demora 100 ns, e assim 500 bits é o menor tamanho de quadro que oferece a garantia de funcionar. Para acrescentar uma certa margem de segurança, esse número foi arredondado para 512 bits ou 64 bytes.

O último campo Ethernet é o *Checksum*. Ele é um CRC de 32 bits do tipo que estudamos na Seção 3.2. De fato, ele é definido exatamente pelo polinômio gerador que mostramos lá, que apareceu para PPP, ADSL e outros enlaces também. Esse CRC é um código de detecção de erro usado para determinar se os bits do quadro foram recebidos corretamente. Ele simplesmente realiza a detecção de erros, com o quadro sendo descartado se algum erro for detectado.

CSMA/CD com backoff exponencial binário

A Ethernet clássica utiliza o algoritmo CSMA/CD 1-persistente que estudamos na Seção 4.2. Esse descritor só significa que as estações sentem o meio quando elas têm um quadro para transmitir e o enviam assim que o meio se torna desocupado. Elas monitoram o canal em busca de colisões enquanto transmitem. Se houver uma colisão, elas cancelam a transmissão com um curto sinal de interferência e retransmitem após um intervalo aleatório.

Vejamos agora como é determinado o intervalo aleatório quando ocorre uma colisão, pois esse é um método novo. O modelo ainda é o da Figura 4.5. Depois de uma colisão, o tempo é dividido em slots discretos, cujo comprimento é igual ao pior tempo de propagação da viagem de ida e volta no éter (2τ). Para acomodar o caminho mais longo permitido pelo padrão Ethernet, o tempo de duração do slot foi definido como 512 períodos de duração de um bit, ou 51,2 μs.

Depois da primeira colisão, cada estação espera 0 ou 1 tempo de slot aleatoriamente antes de tentar novamente. Se duas estações colidirem e selecionarem o mesmo número aleatório, elas colidirão novamente. Depois da segunda colisão, cada uma seleciona ao acaso 0, 1, 2 ou 3 e aguarda durante esse número de tempos de slot. Se ocorrer uma terceira colisão (cuja probabilidade é de 0,25), na próxima vez o número de slots que a estação deverá esperar será escolhido ao acaso no intervalo de 0 a $2^3 - 1$.

Em geral, depois de i colisões, é escolhido um número aleatório entre 0 e $2^i - 1$, e esse número de slots será ignorado. Entretanto, após terem sido alcançadas dez colisões, o intervalo de randomização será congelado em um máximo de 1.023 slots. Depois de 16 colisões, o controlador desiste e informa o erro ao computador. Qualquer recuperação adicional caberá às camadas superiores.

Esse algoritmo, chamado **backoff exponencial binário**, foi escolhido para se adaptar dinamicamente ao número de estações que estão tentando transmitir. Se o intervalo de escolha do número aleatório para todas as colisões fosse 1023, o risco de duas estações colidirem uma segunda vez seria desprezível, mas o tempo de espera médio depois de uma colisão seria de centenas de períodos de slot, ocasionando um atraso significativo. Por outro lado, se cada estação sempre esperasse entre 0 ou 1 slot, e se 100 estações tentassem transmitir ao mesmo tempo, elas colidiriam repetidas vezes até que 99 delas escolhessem 1 e a estação restante escolhesse 0. Isso poderia levar anos. Aumentando-se exponencialmente o intervalo de tempo aleatoriamente, à medida que ocorre um número cada vez maior de colisões consecutivas, o algoritmo assegura um baixo atraso quando apenas algumas estações colidem, mas também garante que a colisão seja resolvida em um período de tempo razoável quando muitas estações colidirem. A restrição de recuo a 1023 impede que o limite aumente demais.

Se não houver colisão, o transmissor considera que o quadro provavelmente foi entregue com êxito. Ou seja, nem CSMA/CD nem Ethernet oferecem confirmações. Essa escolha é apropriada para canais com fio e de fibra óptica, que possuem baixas taxas de erro. Quaisquer erros que ocorram devem então ser detectados pelo CRC e recuperados pelas camadas mais altas. Para canais sem fios, que possuem mais erros, veremos que as confirmações realmente são utilizadas.

4.3.3 Desempenho da Ethernet

Agora, vamos examinar rapidamente o desempenho da Ethernet sob condições de carga alta e constante, ou seja, k estações sempre prontas a transmitir. Uma análise completa do algoritmo de backoff exponencial binário é muito complicada. Em vez disso, seguiremos Metcalfe e Boggs (1976) e vamos supor uma probabilidade de retransmissão constante em cada slot. Se cada estação transmitir durante um slot de disputa com probabilidade p, a probabilidade A de que alguma estação tome posse do canal existente nesse slot será:

$$A = kp(1-p)^{k-1} \tag{4.5}$$

A é maximizado quando $p = 1/k$, com $A \rightarrow 1/e$, à medida que $k \rightarrow \infty$. A probabilidade de que o intervalo de disputa tenha exatamente j slots é $A(1-A)^{j-1}$, de forma que o número médio de slots por disputa é dado por

$$\sum_{j=0}^{\infty} jA(1-A)^{j-1} = \frac{1}{A}$$

Como cada slot tem uma duração de 2τ, o intervalo médio de disputa, w, é $2\tau/A$. Supondo-se um valor ideal para p, o número médio de slots de disputa nunca será maior que 'e'; portanto, w é, no máximo, $2\tau e \approx 5{,}4\tau$.

Se o quadro leva em média P segundos para ser transmitido, quando muitas estações tiverem quadros para enviar,

$$\text{Eficiência do canal} = \frac{P}{P + 2\tau / A} \tag{4.6}$$

Aqui, vemos que a distância máxima do cabo entre duas estações entra nos números do desempenho. Quanto

maior for o cabo, maior será o intervalo de disputa, o que explica por que o padrão Ethernet especifica um comprimento máximo de cabo.

É instrutivo formular a Equação 4.6 em termos do comprimento do quadro, *F*, da largura de banda da rede, *B*, do comprimento do cabo, *L*, e da velocidade de propagação do sinal, *c*, para o caso ideal de '*e*' slots de disputa por quadro. Com $P = F/B$, a Equação 4.6 passa a ser:

$$\text{Eficiência do canal} = \frac{1}{1 + 2BLe / cF} \quad (4.7)$$

Quando o segundo termo no denominador for grande, a eficiência da rede será baixa. Mais especificamente, aumentar a largura de banda de rede ou a distância (o produto *BL*) reduz a eficiência para determinado tamanho de quadro. Infelizmente, a maior parte das pesquisas em hardware de rede visa exatamente ao aumento desse produto. As pessoas querem alta largura de banda a longas distâncias (MANs de fibra óptica, por exemplo), o que sugere que o padrão Ethernet implementado dessa maneira talvez não seja o melhor sistema para essas aplicações. Veremos outras formas de implementar a Ethernet na próxima seção.

Na Figura 4.16, a eficiência do canal é representada contra o número de estações prontas para $2\tau = 51{,}2\ \mu s$ e uma taxa de dados de 10 Mbps, usando-se a Equação 4.7. Com um tempo por slot de 64 bytes, não surpreende que quadros de 64 bytes não sejam eficientes. Por outro lado, com quadros de 1.024 bytes e um valor assintótico de *e* slots de 64 bytes por intervalo de disputa, o período de disputa é de 174 bytes e a eficiência é de 85 por cento. Esse resultado é muito melhor do que os 37 por cento de eficiência da slotted ALOHA.

Talvez valha a pena mencionar que houve um grande número de análises teóricas sobre o desempenho da Ethernet (e de outras redes). A maioria dos resultados deve ser considerada com cautela por duas razões. Primeiro, praticamente todos esses trabalhos presumem que o tráfego obedece a uma série de Poisson. Como os pesquisadores começaram a analisar dados reais, parece que agora o tráfego de rede raras vezes é de Poisson, e sim semelhante ou em forma de rajada por um intervalo de escalas de tempo (Paxson e Floyd, 1995; e Willinger et al., 1994). Isso significa que calcular uma média durante intervalos longos não suaviza o tráfego. Assim como no uso de modelos questionáveis, muitas das análises focam nos casos de desempenho 'específicos' para carga estranhamente alta. Boggs et al. (1988) mostraram, com experimentos, que a Ethernet funciona bem na realidade, até mesmo com carga moderadamente alta.

4.3.4 Ethernet comutada

A Ethernet logo começou a evoluir para longe da arquitetura de cabo longo único da Ethernet clássica (o éter). Os problemas associados a encontrar interrupções ou conexões partidas a levaram para um tipo diferente de padrão de fiação, em que cada estação tem um cabo dedicado esticado até um **hub** central. Um hub simplesmente conecta todos os fios eletricamente, como se eles fossem únicos. Essa configuração pode ser vista na Figura 4.17(a).

Os fios eram pares trançados da companhia telefônica, pois a maioria dos prédios de escritórios já estava conectada dessa forma e normalmente havia muita capacidade ociosa disponível. Esse reúso foi um ganho, mas reduziu o tamanho máximo do cabo do hub para cem metros (duzentos metros, se os pares trançados de alta qualidade da Categoria 5 fossem usados). A inclusão ou remoção de uma estação é mais simples nessa configuração, e uma interrupção de cabo pode ser facilmente detectada. Com a vantagem de elas serem capazes de usar a fiação existente e a facilidade de manutenção, os hubs de par trançado rapidamente se tornaram a forma dominante na topologia Ethernet.

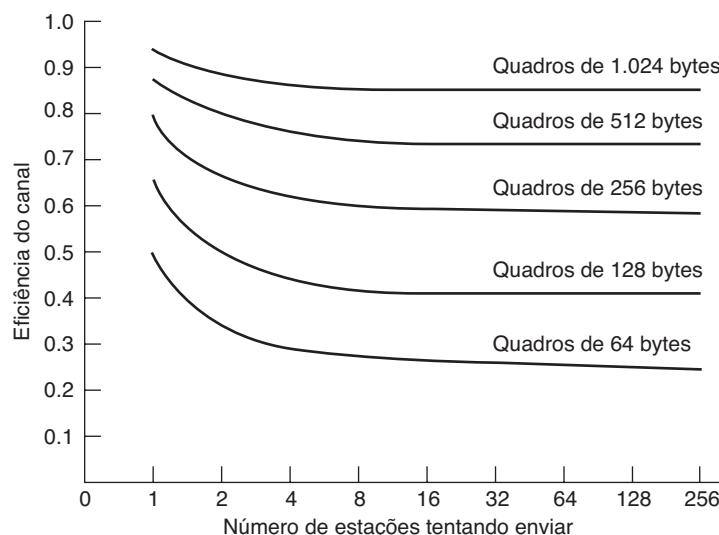



Figura 4.16 | Eficiência da Ethernet a 10 Mbps com tempos por slot de 512 bits.

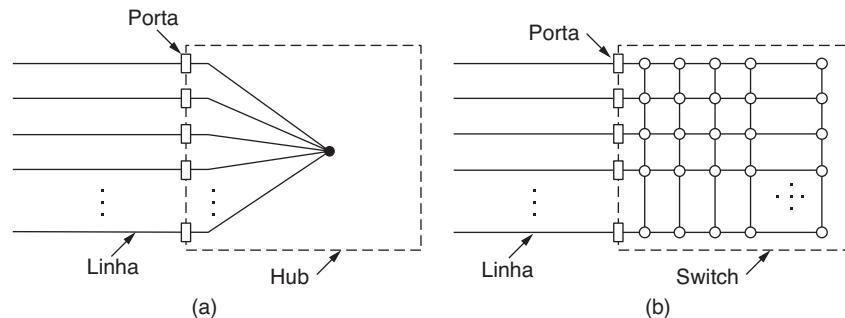



Figura 4.17 | (a) Hub. (b) Switch.

Porém, os hubs não aumentam a capacidade, pois são logicamente equivalentes ao cabo longo único da Ethernet clássica. Quando mais e mais estações são acrescentadas, cada estação recebe uma fatia cada vez menor da capacidade fixa. Por fim, a LAN saturará. Uma saída é usar uma velocidade maior, digamos, de 10 Mbps para 100 Mbps, 1 Gbps ou velocidades ainda maiores. Mas, com o crescimento da multimídia e de servidores poderosos, até mesmo a Ethernet de 1 Gbps pode ficar saturada.

Felizmente, existe outra solução para lidar com o aumento da carga: a Ethernet comutada. O núcleo desse sistema é um **switch**, que contém uma placa integrada (ou *backplane*) de alta velocidade que conecta todas as portas, como mostra a Figura 4.17(b). Por fora, um switch se parece com um hub. Ambos são caixas, normalmente com 4 a 48 portas, cada uma contendo um conector RJ-45 padrão para um cabo de par trançado. Cada cabo conecta o switch ou hub a um único computador, como mostra a Figura 4.18. Um switch também tem as mesmas vantagens de um hub. É muito fácil acrescentar ou remover uma nova estação conectando ou desconectando um fio, e é fácil encontrar a maioria das falhas, pois um cabo ou porta com defeito normalmente afetará apenas uma estação. Ainda existe um componente compartilhado que pode falhar — o próprio switch —, mas, se todas as estações perderem conectividade, o pessoal de TI sabe o que fazer para resolver o problema: trocar o switch inteiro.

Dentro do switch, porém, algo muito diferente está acontecendo. Os switches só enviam quadros às portas para as quais esses quadros são destinados. Quando uma porta do switch recebe um quadro Ethernet de uma estação, o switch verifica os endereços Ethernet para saber para qual porta o quadro se destina. Essa etapa requer que o switch possa descobrir quais portas correspondem a quais endereços, um processo que explicaremos na Seção 4.8, quando analisarmos o caso geral dos switches conectados a outros switches. Por enquanto, basta considerar que o switch conhece a porta de destino do quadro. Depois, o switch encaminha o quadro por sua placa interna de alta velocidade até a porta de destino. A placa interna normalmente trabalha com muitos Gbps, usando um protocolo próprio que não precisa ser padronizado, pois fica inteiramente oculto dentro do switch. A porta de destino, então, transmite o quadro no fio para que ele alcance a estação intencionada. Nenhuma das outras portas sequer saberá que o quadro existe.

O que acontecerá se duas estações ou portas quiserem transmitir um quadro ao mesmo tempo? Novamente, os switches diferem dos hubs. Em um hub, todas as estações estão no mesmo **domínio de colisão**. Elas precisam usar o algoritmo de CSMA/CD para programar suas transmissões. Em um switch, cada porta é seu próprio domínio de colisão independente. No caso comum de um cabo full-duplex, a estação e a porta podem enviar um quadro no cabo ao mesmo tempo, sem se preocupar com outras portas e estações. As colisões agora são impossíveis e o CSMA/CD não é necessário. Porém, se o cabo for half-duplex, a estação e a porta precisam disputar com CSMA/CD pela transmissão, de forma normal.

Um switch melhora o desempenho em relação a um hub de duas maneiras. Primeiro, como não existem colisões, a capacidade é usada de modo mais eficiente. Se-

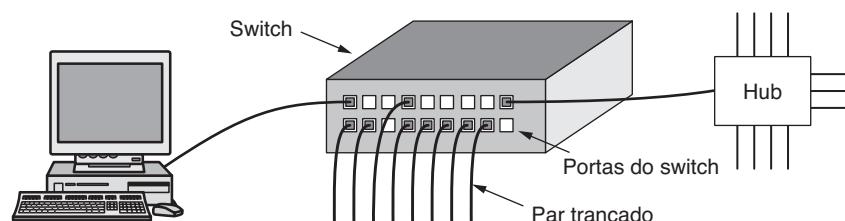



Figura 4.18 | Um switch Ethernet.

gundo, e mais importante, com um switch, vários quadros podem ser enviados simultaneamente (por estações diferentes). Esses quadros alcançarão as portas do switch e trafegarão pela placa integrada do switch para ser enviados nas portas apropriadas. Porém, como dois quadros podem ser enviados para a mesma porta de saída ao mesmo tempo, o switch precisa ter um buffer, para que possa temporariamente enfileirar um quadro de entrada até que ele possa ser transmitido para a porta de saída. Em geral, essas melhorias dão um grande ganho de desempenho, o que não é possível com um hub. O throughput total do sistema normalmente pode ser aumentado em uma ordem de grandeza, dependendo do número de portas e padrões de tráfego.

A mudança nas portas em que os quadros são enviados também tem benefícios para a segurança. A maioria das interfaces de LAN possui um **modo promíscuo**, em que *todos* os quadros são dados a cada computador, não apenas os endereçados a ele. Com um hub, cada computador conectado pode ver o tráfego enviado entre todos os outros computadores. Espiões e bisbilhoteiros adoram esse recurso. Com um switch, o tráfego é encaminhado apenas para as portas às quais ele é destinado. Essa restrição oferece melhor isolamento, de modo que o tráfego não escapará com facilidade nem cairá em mãos erradas. Porém, é melhor criptografar o tráfego se a segurança realmente for necessária.

Tendo em vista que o switch espera apenas quadros Ethernet padrão em cada porta de entrada, é possível usar algumas dessas portas como concentradores. Na Figura 4.18, a porta localizada no canto superior direito não está conectada a uma estação isolada, mas a um hub de 12 portas. À medida que chegam ao hub, os quadros disputam a rede Ethernet de forma normal, inclusive com colisões e backoff exponencial binário. Os quadros bem-sucedidos são enviados ao switch e são tratados como quaisquer outros quadros recebidos. O switch não sabe que eles tiveram de brigar para chegar lá. Uma vez no switch, eles são enviados para a linha de saída correta pela placa integrada de alta velocidade. Também é possível que o destino correto fosse uma das linhas conectadas ao hub, quando o quadro já foi entregue, de modo que o switch simplesmente o descarta. Os hubs são mais simples e mais baratos que os switches, mas, em decorrência da queda nos preços dos switches, eles estão rapidamente se tornando espécies em extinção. Apesar disso, ainda existem hubs legados.

4.3.5 Fast Ethernet

Ao mesmo tempo em que os switches estavam se tornando populares, a velocidade da Ethernet de 10 Mbps estava sendo pressionada. Em princípio, 10 Mbps parecia ser o paraíso, da mesma forma que os modems a cabo pareciam ser o paraíso para os usuários de modems telefônicos. Porém, a novidade se dissipou com rapidez. Como uma espécie de corolário da Lei de Parkinson ("O trabalho se expande até preencher o tempo disponível para sua conclusão"), parecia que os dados se expandiam para preencher toda a largura de banda disponível para sua transmissão.

Muitas instalações precisavam de maior largura de banda e tinham diversas LANs de 10 Mbps conectadas por um labirinto de repetidores, hubs e switches, embora às vezes parecesse, para os administradores de redes, que elas estavam conectadas por goma de mascar e tela de arame. Porém, até mesmo com switches Ethernet, a largura de banda máxima de um único computador era limitada pelo cabo que o conectava à porta do switch.

Foi nesse ambiente que o IEEE reuniu o comitê do 802.3 em 1992, com instruções para produzir uma LAN mais rápida. Uma das propostas era manter o 802.3 exatamente como estava, apenas tornando-o mais rápido. Outra proposta era refazê-lo completamente, para integrar um grande número de novos recursos, como tráfego em tempo real e voz digitalizada, mas manter o antigo nome (por motivos de marketing). Após alguma discussão, o comitê decidiu manter o 802.3 como ele era, simplesmente tornando-o mais rápido. Essa estratégia realizaria o trabalho antes que a tecnologia mudasse, evitando problemas não previstos com um projeto totalmente novo. O novo projeto também seria compatível com as LANs Ethernet existentes. As pessoas que apoiavam a proposta perdedora fizeram o que qualquer pessoa do setor de informática que se preza faria nessas circunstâncias — formaram seu próprio comitê e padronizaram sua LAN assim mesmo (como o padrão 802.12). Esse padrão fracassou por completo.

O trabalho foi feito rapidamente (pelas normas dos comitês de padronização) e o resultado, o 802.3u, foi oficialmente aprovado pelo IEEE em junho de 1995. Tecnicamente, o 802.3u não é um padrão novo, mas um adendo ao padrão 802.3 existente (para enfatizar sua compatibilidade). Como praticamente todos o chamam de **Fast Ethernet**, em vez de 802.3u, faremos o mesmo.

A ideia básica por trás da Fast Ethernet era simples: manter os antigos formatos de quadros, interfaces e regras de procedimentos, e apenas reduzir o tempo de bit de 100 ns para 10 ns. Tecnicamente, teria sido possível copiar a Ethernet clássica de 10 Mbps e continuar a detectar colisões a tempo, pela simples redução do comprimento máximo do cabo a um décimo do comprimento original. Entretanto, as vantagens do cabeamento de par trançado eram tão grandes que a Fast Ethernet se baseou inteiramente nesse projeto. Por isso, todos os sistemas Fast Ethernet usam hubs e switches; porém, cabos multiponto com conectores de pressão ou conectores BNC não são mais permitidos.

Entretanto, algumas decisões ainda precisavam ser tomadas, sendo que a mais importante dizia respeito aos tipos de fios que seriam aceitos. Um dos concorrentes era o par trançado da Categoria 3. O argumento a favor dele era que todo escritório do mundo ocidental tinha pelo menos qua-

tro pares trançados da Categoria 3 (ou melhor) instalados entre ele e um armário de fiação telefônica a uma distância máxima de cem metros. Às vezes, há dois cabos desse tipo. Desse modo, o uso do par trançado da Categoria 3 tornaria possível conectar computadores desktop com o emprego de Fast Ethernet, sem a necessidade de refazer a fiação do edifício, uma enorme vantagem para muitas empresas.

A principal desvantagem do par trançado da Categoria 3 é sua incapacidade para transportar sinais de 100 Mbps por cem metros, a distância máxima especificada entre o computador e o hub para hubs de 10 Mbps. Por outro lado, a fiação de par trançado da Categoria 5 é capaz de tratar cem metros com facilidade, e a fibra pode ir muito mais longe que isso. Decidiu-se permitir as três possibilidades, como mostra a Tabela 4.1, mas incentivar a solução da Categoria 3, para que fosse possível obter a capacidade de transporte adicional necessária.

O esquema de par trançado sem blindagem, ou UTP (*Unshielded Twisted Pair*), da Categoria 3, chamado **100Base-T4**, emprega uma velocidade de sinalização de 25 MHz, somente 25 por cento mais rápida do que os 20 MHz da Ethernet padrão. (Lembre-se de que a codificação Manchester, discutida na Seção 2.5, requer dois períodos de clock para cada um dos 10 milhões de bits enviados a cada segundo.) Porém, para atingir a largura de banda necessária, o 100Base-T4 exige quatro pares trançados. Dos quatro pares, um sempre é para o hub, um sempre é do hub e os outros dois são comutáveis para a direção da transmissão atual. Para conseguir 100 Mbps dos três pares trançados na direção da transmissão, um esquema bastante complicado é usado em cada par trançado. Ele envolve o envio de dígitos ternários com três níveis de tensão. Esse esquema provavelmente não ganhará nenhum prêmio de elegância, e deixaremos de lado os detalhes. Porém, como a fiação da telefonia-padrão há décadas tem quatro pares por cabo, a maioria dos escritórios é capaz de usar a fiação existente. É claro que isso significa abrir mão do telefone do seu escritório, mas esse certamente é um pequeno preço a pagar por um e-mail mais rápido.

O 100Base-T4 foi deixado de lado quando muitos prédios de escritórios tiveram a fiação trocada para o UTP de Categoria 5 para Ethernet **100Base-TX**, que veio para dominar o mercado. Esse projeto é mais simples porque os fios podem lidar com taxas de clock de 125 MHz. Somente dois pares trançados por estação são usados, um para o hub e outro a partir dele. Nem a codificação binária direta (ou seja, NRZ) nem a codificação Manchester são usadas. Em vez disso, é usada a codificação **4B/5B**, que descrevemos na Seção 2.5. Quatro bits de dados são codificados como 5 bits de sinal e enviados a 125 MHz para fornecer 100 Mbps. Esse esquema é simples, mas tem transições suficientes para sincronização e usa a largura de banda do fio relativamente bem. O sistema 100Base-TX é full-duplex; as estações podem transmitir a 100 Mbps em um par trançado e recebem em 100 Mbps em outro par trançado ao mesmo tempo.

A última opção, o **100Base-FX**, utiliza dois filamentos de fibra multimodo, um para cada sentido; por isso, ele também é full-duplex, com 100 Mbps em cada sentido. Nessa configuração, a distância entre uma estação e o switch pode ser de até 2 Km.

A Fast Ethernet permite a interconexão por hubs ou switches. Para garantir que o algoritmo CSMA/CD continue a funcionar, o relacionamento entre o tamanho de quadro mínimo e o tamanho de cabo máximo deve ser mantido enquanto a velocidade da rede sobe de 10 Mbps para 100 Mbps. Assim, ou o comprimento mínimo do quadro de 64 bytes deve aumentar ou o comprimento de cabo máximo de 2.500 m deve diminuir proporcionalmente. A escolha fácil foi que a distância máxima entre duas estações quaisquer fosse diminuída por um fator de 10, pois um hub com cabos de 100 m já está dentro desse novo máximo. Contudo, os cabos 100Base-FX de 2 km são muito longos para aceitar um hub de 100 Mbps com o algoritmo de colisão normal da Ethernet. Esses cabos, em vez disso, precisam ser conectados a um switch e operar em um modo full-duplex, para que não haja colisões.

Os usuários rapidamente começaram a implantar a Fast Ethernet, mas eles não quiseram abandonar as placas Ethernet de 10 Mbps nos computadores mais antigos. Por conseguinte, praticamente todos os switches Ethernet podem lidar com uma mistura de estações de 10 Mbps e 100 Mbps. Para facilitar o upgrading, o próprio padrão oferece um mecanismo chamado **autonegociação**, que permite que duas estações negociem automaticamente a velocidade ideal (10 ou 100 Mbps) e o tipo de duplex (half ou full). Isso quase sempre funciona bem, mas pode ocasionar problemas de divergência do duplex quando uma extremidade do enlace autonegocia e a outra não, ficando definida como o modo full-duplex (Shalunov e Carlson, 2005). A maioria dos produtos Ethernet utiliza esse recurso para se configurar.

Nome	Cabo	Tam. máx. de segmento	Vantagens
100Base-T4	Par trançado	100 m	Utiliza UTP da Categoria 3
100Base-TX	Par trançado	100 m	Full-duplex a 100 Mbps (UTP da Categoria 5)
100Base-FX	Fibra óptica	2.000 m	Full-duplex a 100 Mbps; grandes distâncias

Tabela 4.1 | O cabeamento Fast Ethernet original.

4.3.6 Gigabit Ethernet

A tinta mal havia secado no padrão Fast Ethernet quando o comitê 802 começou a trabalhar em uma Ethernet ainda mais rápida, prontamente apelidada de **gigabit Ethernet**. O IEEE ratificou a forma mais popular como 802.3ab em 1999. A seguir descreveremos algumas das principais características da gigabit Ethernet. Você poderá encontrar mais informações em Spurgeon (2000).

Os objetivos do comitê para a gigabit Ethernet eram essencialmente os mesmos do comitê para a Fast Ethernet: tornar a Ethernet dez vezes mais rápida, mantendo a compatibilidade com todos os padrões Ethernet existentes. Em particular, a gigabit Ethernet tinha de oferecer o serviço de datagrama não confirmado com unicasting e multicasting, empregar o mesmo esquema de endereçamento de 48 bits já em uso e manter o mesmo formato de quadro, inclusive seus tamanhos mínimo e máximo. O padrão final atendeu a todos esses objetivos.

Também como a Fast Ethernet, todas as configurações da gigabit Ethernet utilizam enlaces ponto a ponto. Na configuração mais simples, ilustrada na Figura 4.19(a), dois computadores estão diretamente conectados um ao outro. Porém, o caso mais comum consiste em um switch ou um hub conectado a vários computadores e possivelmente a switches ou hubs adicionais, como mostra a Figura 4.19(b). Em ambas as configurações, cada cabo Ethernet tem exatamente dois dispositivos conectados a ele, nem mais nem menos.

Também como a Fast Ethernet, a gigabit Ethernet admite dois modos de operação: o modo full-duplex e o modo half-duplex. O modo 'normal' é o full-duplex, que permite tráfego em ambos os sentidos ao mesmo tempo. Esse modo é usado quando existe um switch central conectado a computadores (ou outros switches) na periferia. Nessa configuração, todas as portas têm buffers de armazenamento, de forma que cada computador e cada switch são livres para enviar quadros sempre que quiserem. O transmissor não precisa detectar o canal para saber se ele está sendo usado por mais alguém, pois a disputa é impossível. Na linha entre um computador e um switch, o computador é o único transmissor possível para o switch naquela linha, e a transmissão terá sucesso ainda que o switch nesse instante esteja transmitindo um quadro para o computador (porque a linha é full-duplex). Tendo em vista que nenhuma disputa é possível, o protocolo CSMA/CD não é usado, e assim o comprimento máximo do cabo é determinado pela intensidade do sinal, não pelo tempo que uma rajada de sinal ruidoso leva para se propagar de volta até o transmissor na pior das hipóteses. Os switches são livres para se misturar e combinar suas velocidades. A autonegociação é admitida, como na Fast Ethernet, mas agora a escolha é entre 10, 100 e 1.000 Mbps.

O outro modo de operação, o half-duplex, é usado quando os computadores estão conectados a um hub, não a um switch. Um hub não armazena os quadros recebidos em buffers. Em vez disso, ele estabelece conexões elétricas internas para todas as linhas, simulando o cabo multiponto usado na Ethernet clássica. Nesse modo, colisões são possíveis e, portanto, é necessário o protocolo CSMA/CD-padrão. Tendo em vista que um quadro mínimo de 64 bytes (o mais curto permitido) agora pode ser transmitido cem vezes mais rápido que na Ethernet clássica, a distância máxima é cem vezes menor (ou seja, 25 metros), a fim de manter a propriedade essencial de que o transmissor ainda transmitirá quando uma rajada de sinal ruidoso voltar a ele, mesmo na pior das hipóteses. Com um cabo de 2.500 metros, o transmissor de um quadro de 64 bytes a 1 Gbps terminaria a transmissão bem antes de o quadro sequer ter chegado a percorrer um décimo da distância até a outra extremidade, quanto mais ir até a extremidade e voltar.

Essa restrição de distância foi tão séria que duas características foram acrescentadas ao padrão para aumentar a distância máxima do cabo para duzentos metros, o que provavelmente é suficiente para a maioria dos escritórios. A primeira característica, chamada **extensão de portadora**, essencialmente informa ao hardware para adicionar seu próprio preenchimento ao quadro normal, a fim de estendê-lo para 512 bytes. Tendo em vista que esse preenchimento é adicionado pelo hardware transmissor e removido pelo hardware receptor, o software não tem conhecimento desse fato, o que significa que não é necessária nenhuma

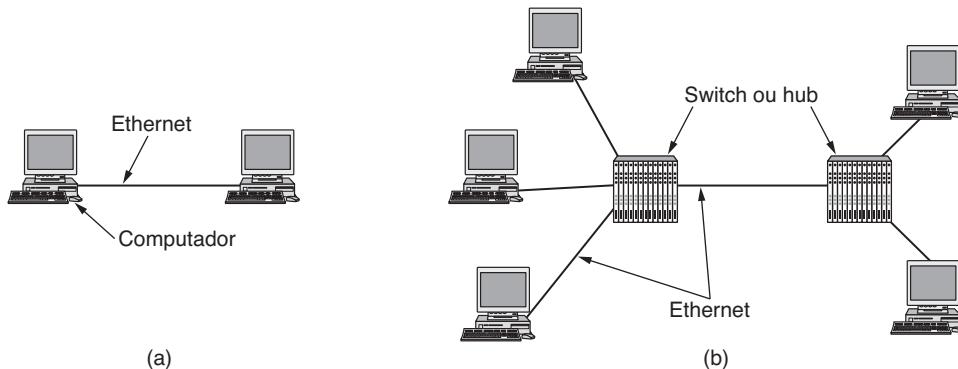



Figura 4.19 | (a) Uma Ethernet com duas estações. (b) Uma Ethernet com várias estações.

mudança no software existente. É claro que o uso de 512 bytes de largura de banda para transmitir 46 bytes de dados do usuário (a carga útil de um quadro de 64 bytes) tem uma eficiência de linha de apenas 9 por cento.

A segunda característica, chamada **rajada de quadros**, permite a um transmissor enviar uma sequência concatenada de vários quadros em uma única transmissão. Se a rajada total tiver menos de 512 bytes, o hardware a preencherá novamente. Se houver quadros suficientes esperando pela transmissão, esse esquema será altamente eficiente e preferível à extensão de portadora.

Com toda franqueza, é difícil imaginar uma organização se envolvendo com as dificuldades de compra e instalação de placas gigabit Ethernet para obter alto desempenho, e depois conectar os computadores a um antigo hub para simular a Ethernet clássica, com todas as suas colisões. As interfaces e os switches da gigabit Ethernet eram muito caros, mas seu preço caiu rapidamente à medida que o volume de vendas aumentou. Ainda assim, a compatibilidade é sagrada na indústria de informática e, então, o comitê é obrigado a aceitá-la. Hoje, a maioria dos computadores vem com uma interface Ethernet capaz de operar a 10, 100 e 1.000 Mbps, compatível com todas as velocidades.

A gigabit Ethernet admite cabeamento de cobre e de fibra, como mostra a Tabela 4.2. A sinalização à velocidade de aproximadamente 1 Gbps requer a codificação e o envio de um bit a cada nanossegundo. Esse truque foi realizado inicialmente com cabos de cobre curtos e blindados (a versão 1000Base-CX) e fibras ópticas. Para estas, dois comprimentos de onda são permitidos e o resultado são duas versões: 0,85 mícron (curto, para 1000Base-SX) e 1,3 mícron (longo, para 1000Base-LX).

A sinalização no comprimento de onda curto pode ser alcançada com LEDs mais baratos. Ela é usada com a fibra multimodo e é útil para conexões dentro de um prédio, pois pode se estender até 500 m para a fibra de 50 micra. A sinalização no comprimento de onda longo exige lasers mais caros. Por outro lado, quando combinado com a fibra de modo único (10 micra), o comprimento do cabo pode ser de até 5 Km. Esse limite permite conexões a longa distância entre prédios, como para o backbone de um campus, como em um enlace ponto a ponto dedicado. Outras variações do padrão permitiram enlaces ainda mais longos sobre a fibra de modo único.

Para enviar bits por essas versões da gigabit Ethernet, a codificação **8B/10B**, que descrevemos na Seção 2.5, foi emprestada de outra tecnologia de redes, chamada Fibre Channel. Esse esquema codifica 8 bits de dados em dez palavras de código de 10 bits, que são enviadas pelo fio ou fibra, daí o nome 8B/10B. As palavras de código foram escolhidas de modo que pudessem ser balanceadas (ou seja, tivessem o mesmo número de 0s e 1s) com transições suficientes para a recuperação de clock. O envio dos bits codificados com NRZ requer uma largura de banda de sinalização de 25 por cento a mais do que a necessária para os bits não codificados, uma grande melhoria em relação à expansão de 100 por cento da codificação Manchester.

Contudo, todas essas opções exigiam novos cabos de cobre ou fibra para dar suporte à sinalização mais rápida. Nenhum deles utilizava a grande quantidade de UTP de Categoria 5 que havia sido instalada com a Fast Ethernet. Dentro de um ano, o 1000Base-T surgiu para preencher essa lacuna, e tem sido a forma mais popular de gigabit Ethernet desde então. As pessoas aparentemente não quiseram mudar a fiação de seus prédios.

Uma sinalização mais complicada é necessária para fazer a Ethernet funcionar a 1.000 Mbps sobre fios de Categoria 5. Para começar, todos os quatro pares trançados no cabo são usados, e cada par é usado nas duas direções ao mesmo tempo, usando o processamento de sinal digital para separar os sinais. Pelos fios, um a um, cinco níveis de voltagem que transportam 2 bits são usados para sinalizar em 125 Msímbolos/s. O mapeamento para produzir os símbolos a partir dos bits não é simples. Ele envolve embaralhamento e transições, seguidos por um código de correção de erros em que quatro valores são embutidos em cinco níveis de sinal.

Uma velocidade de 1 Gbps é bastante alta. Por exemplo, se um receptor estiver ocupado com alguma outra tarefa, mesmo durante 1 ms, e não esvaziar o buffer de entrada em alguma porta, nesse intervalo poderão se acumular até 1.953 quadros. Além disso, quando um computador em uma gigabit Ethernet estiver transmitindo dados pela linha a um computador em uma Ethernet clássica, serão muito prováveis sobrecargas no buffer. Como consequência dessas duas observações, a gigabit Ethernet admite controle de fluxo. O mecanismo consiste na transmissão de um quadro de controle especial de uma extremidade a ou-

Nome	Cabo	Distância máxima do segmento	Vantagens
1000Base-SX	Fibra óptica	550 m	Fibra multimodo (50, 62,5 micra)
1000Base-LX	Fibra óptica	5.000 m	Modo único (10 micra) ou multimodo (50, 62,5 micra)
1000Base-CX	2 pares de STP	25 m	Par trançado blindado
1000Base-T	4 pares de UTP	100 m	UTP padrão da Categoria 5

Tabela 4.2 | O cabeamento da gigabit Ethernet.

tra, informando que a extremidade receptora deve fazer uma pausa durante algum período predeterminado. Para controle de fluxo, são usados quadros PAUSE, contendo o tipo 0x8808. As pausas são dadas em unidades de tempo mínimo por quadro. Para a gigabit Ethernet, a unidade de tempo é 512 ns, permitindo pausas de até 33,6 ms.

Existe mais uma extensão introduzida com a gigabit Ethernet. **Quadros jumbo** permitem que os quadros tenham mais de 1.500 bytes, normalmente até 9 KB. Essa extensão é patenteada. Ela não é reconhecida pelo padrão porque, se for usada, a Ethernet não será mais compatível com versões anteriores, mas, de qualquer forma, a maioria dos vendedores oferece suporte para ela. O raciocínio é que 1.500 bytes representam uma unidade curta nas velocidades de gigabit. Manipulando blocos de informação maiores, a taxa de quadros pode ser diminuída, com o processamento associado a ela, como a interrupção do processador para dizer que um quadro chegou, ou a divisão e recombinação de mensagens que eram muito grandes para caber em um quadro Ethernet.

4.3.7 Ethernet de 10 gigabits

Assim que a gigabit Ethernet foi padronizada, o comitê 802 ficou entediado e quis voltar ao trabalho. O IEEE pediu que se iniciassem os estudos sobre a Ethernet de 10 gigabits. Esse trabalho seguiu aproximadamente os mesmos moldes dos padrões Ethernet anteriores, com os padrões para fibra e cabo de cobre blindado, que apareceram primeiro em 2002 e 2004, seguidos pelo padrão para par trançado de cobre em 2006.

A velocidade de 10 Gbps é verdadeiramente prodigiosa, mil vezes mais rápida que a Ethernet original. Onde ela poderia ser necessária? A resposta é: dentro dos centros de dados e estações, para conectar roteadores de ponta, switches e servidores, bem como em troncos de longa distância e alta largura de banda, entre estações que habilitam redes metropolitanas inteiras com base em Ethernet e fibra. As conexões de longa distância utilizam fibra óptica, enquanto as conexões curtas podem usar cobre ou fibra.

Todas as versões da Ethernet de 10 gigabits admitem apenas operação full-duplex. CSMA/CD não faz mais parte do projeto, e os padrões se concentram em detalhes das camadas físicas que podem trabalhar em velocidades muito altas. Porém, a compatibilidade ainda é importante, de modo que as interfaces Ethernet de 10 gigabits autonegociam e recuam para a velocidade mais baixa admitida pelas duas extremidades da linha.

Os principais tipos de Ethernet de 10 gigabits são listados na Tabela 4.3. A fibra multimodo com comprimento de onda de 0,85μ (curta) e 1,5μ (estendida) é usada para longas distâncias. A 10GBase-ER pode percorrer distâncias de 40 km, o que a torna adequada para aplicações remotas. Todas essas versões enviam um fluxo serial de informações, produzido pelo embaralhamento dos bits de dados, depois a codificação com um código **64B/66B**. Essa codificação tem menos overhead do que uma codificação 8B/10B.

A primeira versão de cobre definida, 10GBase-CX4, usa um cabo com quatro pares de fiação de cobre twinaxial. Cada par usa codificação 8B/10B e trabalha a 3,125 Gsímbolos/s para alcançar 10 Gbps. Essa versão é mais barata do que a fibra e chegou cedo ao mercado, mas ainda não sabemos se a longo prazo vencerá a Ethernet de 10 gigabits sobre a fiação de par trançado de variedade comum.

A 10GBase-T é uma versão que usa cabos UTP. Embora exija fiação de Categoria 6a, para pequenas distâncias, ela pode usar categorias inferiores (incluindo a Categoria 5) para permitir algum reúso do cabeamento instalado. Não é surpresa que a camada física seja muito complicada para alcançar 10 Gbps sobre par trançado. Só veremos por alto alguns dos detalhes de alto nível. Cada um dos quatro pares trançados é usado para enviar 2.500 Mbps/s em ambas as direções. Essa velocidade é alcançada usando-se uma taxa de sinalização de 800 Msímbolos/s, com símbolos que usam 16 níveis de tensão. Os símbolos são produzidos embaralhando-se os dados, protegendo-os com o código LDPC (Low Density Parity Check) e codificando ainda mais para correção de erro.

A Ethernet de 10 gigabits ainda está sacudindo o mercado, mas o comitê 802.3 já fez progressos. Ao final de 2007, o IEEE criou um grupo para padronizar a Ethernet operando a 40 Gbps e 100 Gbps. Essa atualização permitirá que a Ethernet concorra em ambientes de desempenho muito alto, incluindo conexões de longa distância em redes de backbone e conexões curtas, nas placas internas de equipamentos. O padrão ainda não está concluído, mas já estão disponíveis produtos patenteados.

Nome	Cabo	Distância máxima do segmento	Vantagens
10GBase-SR	Fibra óptica	Até 300 m	Fibra multimodo (0,85 μ)
10GBase-LR	Fibra óptica	10 Km	Fibra monomodo (1,3 μ)
10GBase-ER	Fibra óptica	40 Km	Fibra monomodo (1,5 μ)
10GBase-CX4	4 pares de twinax	15 m	Cobre twinaxial
10GBase-T	4 pares de UTP	100 m	UTP padrão da Categoria 6a

Tabela 4.3 | O cabeamento da Ethernet de 10 gigabits.

4.3.8 Retrospectiva da Ethernet

A Ethernet existe há mais de trinta anos e não tem concorrentes sérios; portanto, é provável que continue no mercado ainda por muitos anos. Poucas arquiteturas de CPUs, sistemas operacionais ou linguagens de programação seriam capazes de se manter na liderança por três décadas, continuando com força. Sem dúvida, a Ethernet fez algo correto. O que foi?

Provavelmente a principal razão para sua longevidade seja o fato de que a Ethernet é simples e flexível. Na prática, simples se traduz como confiável, de baixo custo e de fácil manutenção. Depois que a arquitetura de hub e switch foi adotada, as falhas se tornaram extremamente raras. As pessoas hesitam em substituir algo que funciona bem o tempo todo, em especial quando sabem que uma quantidade terrível de itens da indústria de informática funciona muito mal. Muitas das chamadas 'atualizações' são bem piores que as versões substituídas por elas.

Simplicidade também se traduz em economia. A fiação de par trançado tem custo relativamente baixo, assim como os componentes do hardware. Eles começam caros quando há uma transição, por exemplo, novas NICs ou switches da gigabit Ethernet, mas são apenas acréscimos a uma rede bem estabelecida (não uma substituição dela) e os preços caem rapidamente à medida que o volume de vendas aumenta.

A Ethernet é de fácil manutenção. Não existe nenhum software para instalar (além dos drivers) e não há nenhuma tabela de configuração para gerenciar (e errar). Além disso, a inclusão de novos hosts é simples: basta conectá-los.

Outro ponto importante é que a Ethernet é capaz de interoperar facilmente com o TCP/IP, que se tornou dominante. O IP é um protocolo não orientado a conexões e, portanto, se ajusta perfeitamente à Ethernet, que também é não orientada a conexões. O IP não tem a mesma facilidade para se ajustar a alternativas orientadas a conexão, como o ATM. Essa falta de compatibilidade definitivamente diminui as chances de sucesso do ATM.

Por fim, e talvez mais importante, a Ethernet foi capaz de evoluir em certos aspectos cruciais. As velocidades aumentaram várias ordens de magnitude, e os hubs e switches foram introduzidos, mas essas mudanças não exigiram alterações no software e normalmente permitiam que o cabeamento existente fosse reutilizado por um tempo. Quando um vendedor de redes aparece em uma grande instalação e diz: "Tenho esta nova e fantástica rede para você. Basta se desfazer de todo o seu hardware e reescrever todo o seu software", ele tem um problema.

Muitas tecnologias alternativas, que você provavelmente nem sequer ouviu falar, eram mais rápidas que a Ethernet quando foram introduzidas. Assim como o ATM, essa lista inclui o FDDI (Fiber Distributed Data Interface) e o Fibre Channel,† duas LANs ópticas baseadas em anéis. Ambas eram incompatíveis com a Ethernet. Nenhuma delas teve sucesso. Elas eram muito complicadas, o que resultava em chips complexos e altos preços. A lição que deveria ter sido aprendida aqui é que as coisas precisam ser mantidas simples. Por fim, a Ethernet os alcançou em termos de velocidade, geralmente pegando parte de sua tecnologia emprestada, por exemplo, a codificação 4B/5B do FDDI e a codificação 8B/10B do Fibre Channel. Então, ou elas não tinham mais vantagens e desapareceram silenciosamente ou tiveram funções especializadas.

Parece que a Ethernet continuará a se expandir em suas aplicações ainda por algum tempo. A Ethernet de 10 gigabits acabou com as restrições de distância do CSMA/CD. Tem sido realizado muito esforço para a **carrier-grade Ethernet** ou simplesmente **Carrier-Ethernet**, para permitir que os provedores de rede ofereçam serviços baseados em Ethernet aos seus clientes para redes metropolitanas e a longas distâncias (Fouli e Maler, 2009). Essa aplicação transporta quadros Ethernet a longas distâncias através da fibra e exige melhores recursos de gerenciamento para ajudar as operadoras a oferecer serviços confiáveis e de alta qualidade. As redes com velocidade muito alta também estão sendo usadas em placas integradas, conectando componentes em grandes roteadores ou servidores. Esses dois usos são adicionais ao envio de quadros entre computadores em escritórios.

4.4 LANs sem fios

As LANs sem fios estão cada vez mais populares e um número crescente de edifícios de escritórios, aeroportos e outros lugares públicos está sendo equipado com elas, para conectar computadores, PDAs e smartphones à Internet. As LANs sem fios também podem ser usadas para permitir que dois ou mais computadores vizinhos se comuniquem sem usar a Internet.

O principal padrão de LAN sem fio é o 802.11. Vimos algumas informações básicas sobre ele na Seção 1.5.3. Agora, vamos examinar mais de perto a tecnologia. Nas próximas seções, estudaremos a pilha de protocolos, as técnicas de transmissão de rádio na camada física, o protocolo da subcamada MAC, a estrutura de quadro e os serviços fornecidos. Para obter mais informações sobre o 802.11, consulte Gast (2005). Para conhecer os detalhes mais profundos, consulte o próprio padrão publicado, o IEEE 802.11-2007.

4.4.1 802.11: arquitetura e pilha de protocolos

As redes 802.11 podem ser usadas em dois modos. O modo mais popular é conectar clientes, como laptops e smartphones, a outra rede, como uma intranet da empresa ou a Internet. Esse modo aparece na Figura 4.20(a). No

† Ele se chama 'Fibre Channel' e não 'Fiber Channel', pois o editor do documento era britânico.

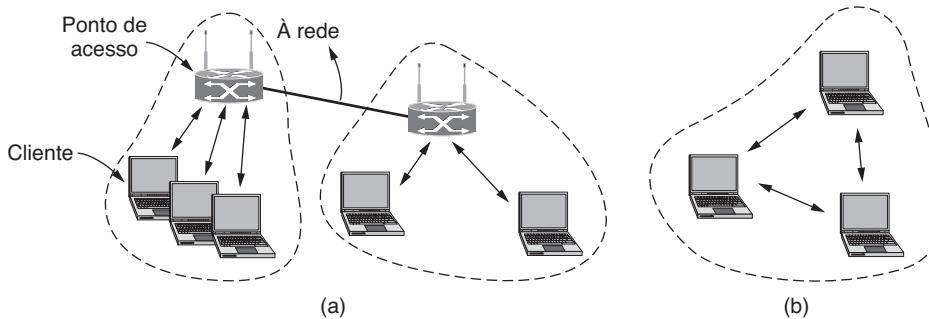



Figura 4.20 | Arquitetura 802.11. (a) Modo de infraestrutura. (b) Modo ad hoc.

modo de infraestrutura, cada cliente está associado a um **PA** (**ponto de acesso**), que, por sua vez, está conectado a outra rede. O cliente transmite e recebe seus pacotes por meio do PA. Vários pontos de acesso podem ser conectados, normalmente por uma rede com fios chamada **sistema de distribuição**, para formar uma rede 802.11 estendida. Nesse caso, os clientes podem enviar quadros aos outros clientes por meio de seus PAs.

O outro modo, mostrado na Figura 4.20(b), é uma **rede ad hoc**. Esse modo é uma coleção de computadores que estão associados de modo que possam enviar quadros diretamente uns aos outros. Não existe ponto de acesso. Como o acesso à Internet é a principal aplicação para redes sem fio, as redes ad hoc não são muito populares.

Agora, vejamos os protocolos. Todos os protocolos 802, incluindo 802.11 e Ethernet, têm certas características comuns em sua estrutura. Uma visão parcial da pilha de protocolos do 802.11 é dada na Figura 4.21. A pilha é a mesma para clientes e PAs. A camada física corresponde muito bem à camada física do modelo OSI, mas a camada de enlace de dados em todos os protocolos 802 se divide em duas ou mais subcamadas. No 802.11, a subcamada MAC (Medium Access Control) determina como o canal é alocado, isto é, quem terá a oportunidade de transmitir a seguir. Acima dela encontra-se a subcamada LLC (Logical Link Control), cujo trabalho é ocultar as diferenças entre as diversas variações do 802 e torná-las indistinguíveis no que se refere à camada de rede. Essa poderia ter sido uma responsividade significativa, mas atualmente a LLC é uma camada de cola, que identifica o protocolo (por exemplo, IP) que é transportado dentro de um quadro 802.11.

Várias técnicas de transmissão foram acrescentadas à camada física à medida que o 802.11 evoluiu desde o seu aparecimento, em 1997. Duas das técnicas iniciais, infravermelho como nos controles remotos de televisão e salto de frequência na banda de 2,4 GHz, agora não são mais usadas. A terceira técnica inicial, o espectro de dispersão de sequência direta a 1 ou 2 Mbps na banda de 2,4 GHz, foi estendida para trabalhar em velocidades de até 11 Mbps e tornou-se rapidamente um sucesso. Ela agora é conhecida como 802.11b.

Para dar aos viciados em redes sem fio o aumento de velocidade tão desejado, novas técnicas de transmissão, baseadas no esquema OFDM (Orthogonal Frequency Division Multiplexing), que descrevemos na Seção 2.5.3, foram introduzidas em 1999 e em 2003. A primeira é chamada 802.11a e usa uma banda de frequência diferente, a de 5 GHz. A segunda ficou com 2,4 GHz e compatibilidade. Ela é chamada 802.11g. Ambas oferecem velocidades de até 54 Mbps.

Mais recentemente, as técnicas de transmissão que usam simultaneamente várias antenas no transmissor e no receptor para aumentar a velocidade foram finalizadas como 802.11n, em outubro de 2009. Com quatro antenas

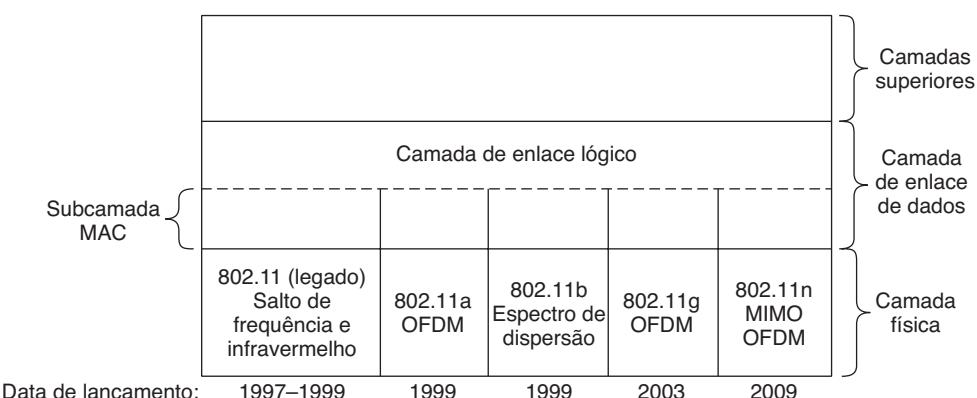



Figura 4.21 | Parte da pilha de protocolos do 802.11.

e canais mais largos, o padrão 802.11 agora define taxas de até incríveis 600 Mbps.

Agora, vamos examinar cada uma dessas técnicas de transmissão em linhas gerais. Porém, abordaremos apenas aquelas que estão em uso, pulando os métodos de transmissão 802.11 legados. Tecnicamente, elas pertencem à camada física e deveriam ter sido examinadas no Capítulo 2; porém, como estão estritamente relacionadas às LANs em geral e em particular à LAN 802.11, preferimos tratá-las aqui.

4.4.2 802.11: A CAMADA FÍSICA

Cada uma das técnicas de transmissão torna possível enviar um quadro MAC de uma estação para outra. Contudo, elas diferem na tecnologia usada e na velocidade que podem alcançar. Uma descrição detalhada dessas tecnologias está muito além do escopo deste livro, mas algumas palavras sobre cada uma relacionarão as técnicas ao conteúdo abordado na Seção 2.5, fornecendo aos leitores interessados material para pesquisar mais informações na Internet ou em outras fontes.

Todas as técnicas do 802.11 utilizam rádios de curto alcance para transmitir sinais nas bandas de frequência ISM de 2,4 GHz ou 5 GHz, ambas descritas na Seção 2.3.3. Essas bandas têm a vantagem de não ser licenciadas e, portanto, estar disponíveis gratuitamente a qualquer transmissor que queira cumprir algumas restrições, como a potência irradiada de no máximo 1 W (embora 50 mW seja mais comum para rádios de LAN sem fios). Infelizmente, esse fato também é conhecido pelos fabricantes de aparelhos de abertura automática de garagem, telefones sem fio, fornos de micro-ondas e diversos outros dispositivos, todos competindo com os notebooks pelo mesmo espectro. A banda de 2,4 GHz costuma ser mais sobrecarregada do que a de 5 GHz, de modo que esta pode ser melhor para algumas aplicações, embora tenha um alcance mais curto, em virtude da frequência mais alta.

Todos os métodos de transmissão também definem taxas múltiplas. A ideia é que diferentes taxas podem ser usadas dependendo das condições atuais. Se o sinal sem fio for fraco, uma taxa baixa poderá ser usada. Se o sinal for claro, a taxa mais alta poderá ser usada. Esses ajustes constituem o que chamamos **adaptação de taxa**. Como as taxas variam por um fator de 10 ou mais, uma boa adaptação de taxa é importante para um bom desempenho. É claro que, pelo fato de ela não ser necessária para a interoperabilidade, os padrões não dizem como a adaptação de taxa deve ser feita.

O primeiro método de transmissão que veremos é o **802.11b**. Ele é um método de espectro de dispersão que admite taxas de 1, 2, 5,5 e 11 Mbps, embora na prática a taxa de operação seja quase sempre 11 Mbps. Isso é semelhante ao sistema CDMA, que examinamos na Seção 2.5, exceto que há somente um código de espalhamento compartilhado por todos os usuários. O espalhamento é usado para satisfazer ao requisito da FCC de que a potência deve ser espalhada pela banda ISM. A sequência de espalhamento usada pelo 802.11b é uma **sequência de Barker**. Ela tem como propriedade a autocorrelação baixa, exceto quando as sequências estão alinhadas. Essa propriedade permite que um receptor intercepte o início de uma transmissão. Para transmitir em uma taxa de 1 Mbps, a sequência de Barker é usada com a modulação BPSK para enviar 1 bit por 11 chips. Os chips são transmitidos a uma taxa de 11 Mchips/s. Para enviar a 2 Mbps, ela é usada com a modulação QPSK para enviar 2 bits por 11 chips. As taxas mais altas são diferentes, pois usam uma técnica conhecida como **chaveamento de código complementar**, ou **CCK** (**Complementary Code Keying**), para construir códigos em vez da sequência de Barker. A taxa de 5,5 Mbps envia 4 bits em cada código de 8 chips, e a taxa de 11 Mbps envia 8 bits em cada código de 8 chips.

Em seguida, chegamos ao **802.11a**, que admite taxas de até 54 Mbps na banda ISM de 5 GHz. Você poderia esperar que o 802.11a viesse antes do 802.11b, mas não foi assim. Embora o grupo 802.11a tenha sido estabelecido primeiro, o padrão 802.11b foi aprovado primeiro e seu produto chegou ao mercado bem antes dos produtos 802.11a, parcialmente em virtude da dificuldade de operar na banda mais alta de 5 GHz.

O método 802.11a é baseado na **multiplexação por divisão ortogonal de frequência**, ou **OFDM** (**Orthogonal Frequency Division Multiplexing**), pois a OFDM usa o espectro com eficiência e resiste a degradações do sinal sem fios, como o enfraquecimento por múltiplos caminhos. Os bits são enviados por 52 subportadoras em paralelo, 48 transportando dados e 4 usadas para sincronização. Cada símbolo dura 4 μs e envia 1, 2, 4 ou 6 bits. Os bits são codificados para correção de erros, com um código de convolução binário, primeiro de modo que somente 1/2, 2/3 ou 3/4 dos bits não são redundantes. Com diferentes combinações, o 802.11a pode trabalhar em oito taxas, variando de 6 a 54 Mbps. Essas taxas são significativamente mais rápidas do que as taxas 802.11b, e existe menos interferência na banda de 5 GHz. Contudo, o 802.11b tem um alcance que é cerca de sete vezes maior que o do 802.11a, o que em muitas situações é mais importante.

Mesmo com o alcance maior, o pessoal do 802.11b não tinha intenção de permitir que esse início vencesse o campeonato de velocidade. Felizmente, em maio de 2002, a FCC retirou sua regra, existente havia muito tempo, de exigir que todo equipamento de comunicação sem fios operasse nas bandas ISM nos Estados Unidos para usar o espectro de dispersão, de modo que passou a trabalhar no **802.11g**, que foi aprovado pelo IEEE em 2003. Ele copia os métodos de modulação OFDM do 802.11a, mas opera na banda ISM estreita de 2,4 GHz, com o 802.11b. Ele oferece as mesmas taxas do 802.11a (6 a 54 Mbps) mais, é claro, a compatibilidade com quaisquer dispositivos 802.11b que estejam nas proximidades. Todas essas diferentes escolhas

podem ser confusas para os clientes, de modo que é comum que os produtos ofereçam suporte para 802.11a/b/g em uma única NIC.

Não satisfeito em parar aí, o comitê do IEEE começou a trabalhar em uma camada física de alto throughput, chamada **802.11n**. Ela foi ratificada em 2009. O objetivo do 802.11n foi um throughput de pelo menos 100 Mbps depois que todos os overheads da rede sem fios fossem removidos. Esse objetivo exigia um aumento de velocidade bruto, com um fator de pelo menos quatro. Para isso acontecer, o comitê dobrou os canais de 20 MHz para 40 MHz e reduziu os overheads de enquadramento, permitindo que um grupo de quadros fosse enviado em conjunto. Porém, o mais significativo é que o 802.11n usa até quatro antenas para transmitir até quatro fluxos de informação ao mesmo tempo. Os sinais dos fluxos interferem no receptor, mas eles podem ser separados usando as técnicas de comunicação de **entrada múltipla, saída múltipla**, ou **MIMO** (**Multiple Input, Multiple Output**). O uso de múltiplas antenas oferece um grande aumento de velocidade e, além disso, melhora o alcance e a confiabilidade. MIMO, como OFDM, é uma daquelas ideias de comunicação inteligentes que estão mudando os projetos das redes sem fios e das quais, provavelmente, todos nós ouviremos falar muito no futuro. Para obter uma breve introdução às antenas múltiplas no 802.11, consulte Halpein et al. (2010).

4.4.3 802.11: o protocolo da subcamada MAC

Agora, vamos retornar dos domínios da engenharia elétrica para os da ciência da computação. O protocolo da subcamada MAC do 802.11 é bastante diferente do protocolo da Ethernet, em razão da complexidade inerente à comunicação sem fio.

Primeiro, os rádios quase sempre são half-duplex, significando que eles não podem transmitir e escutar rajadas de sinais ruidosos ao mesmo tempo em uma única frequência. O sinal recebido pode facilmente ser um milhão de vezes mais fraco do que o sinal transmitido, de modo que não pode ser detectado ao mesmo tempo. Com a Ethernet, uma estação só precisa esperar até o éter ficar inativo para começar a transmitir. Se não receber de volta uma rajada de sinal ruidoso enquanto transmite os primeiros 64 bytes, é quase certo que o quadro tenha sido entregue corretamente. No caso das LANs sem fios, esse mecanismo de detecção de colisão não funciona.

Em vez disso, o 802.11 tenta evitar colisões com um protocolo chamado **CSMA com prevenção de colisão**, ou **CSMA/CA** (**CSMA with Collision Avoidance**). Esse protocolo é conceitualmente semelhante ao CSMA/CD da Ethernet, com detecção de portadora antes de transmitir e o algoritmo de backoff exponencial binário após as colisões. Porém, uma estação que tem um quadro para transmitir começa com um backoff aleatório (exceto no caso em que ela não tenha usado o canal recentemente e o canal esteja inoperante). Ela não espera por uma colisão. O número de slots a recuar é escolhido na faixa de 0 a, digamos, 15, no caso da camada física OFDM. A estação espera até que o canal esteja inoperante, detectando que não existe sinal por um curto período (chamado DIFS, como explicaremos mais adiante), e conta regressivamente os slots inoperantes, interrompendo quando os quadros forem enviados. Ela envia seu quadro quando o contador chega a 0. Se o quadro passar, o destino imediatamente envia uma confirmação curta. A falta de uma confirmação é deduzida como indicativo de erro, seja uma colisão, seja outro erro qualquer. Nesse caso, o transmissor dobra o período de backoff e tenta novamente, continuando com o backoff exponencial binário, como na Ethernet, até que o quadro tenha sido transmitido com sucesso ou o número máximo de retransmissões tenha sido alcançado.

Uma linha do tempo como exemplo aparece na Figura 4.22. A estação *A* é a primeira a transmitir um quadro. Enquanto *A* está transmitindo, as estações *B* e *C* ficam prontas para enviar. Elas veem que o canal está ocupado e esperam até que ele esteja livre. Pouco depois de *A* receber uma confirmação, o canal é liberado. Porém, em vez de enviar um quadro imediatamente e colidir, *B* e *C* realizam

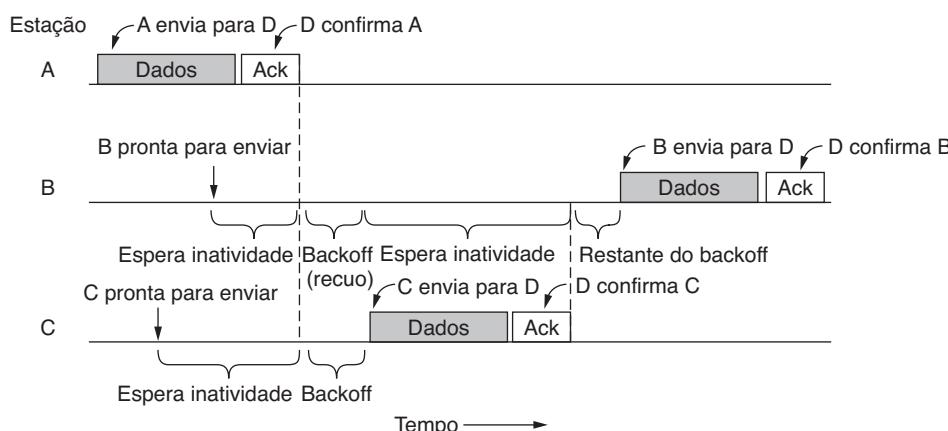



Figura 4.22 | Transmitindo um quadro com CSMA/CA.

um recuo (backoff). *C* escolhe um recuo pequeno e, assim, transmite primeiro. *B* interrompe sua contagem enquanto detecta que *C* está usando o canal e retoma depois que *C* tiver recebido uma confirmação. *B* logo conclui seu recuo e transmite seu quadro.

Em comparação com a Ethernet, existem duas diferenças principais. Primeiro, iniciar os recuos cedo ajuda a evitar colisões. Essa prevenção vale a pena porque as colisões são dispendiosas, já que o quadro inteiro é transmitido mesmo que ocorra uma colisão. Em segundo lugar, as confirmações são usadas para deduzir colisões, pois estas não podem ser detectadas.

Esse modo de operação é chamado **função de coordenação distribuída**, ou **DCF** (**Distributed Coordination Function**), pois cada estação atua de modo independente, sem nenhum tipo de controle central. O padrão também inclui um modo de operação opcional, chamado **função de coordenação de ponto**, ou **PCF** (**Point Coordination Function**), em que o ponto de acesso controla toda a atividade em sua célula, assim como uma estação-base de celular. Porém, o PCF não é usado na prática porque normalmente não existe um modo de impedir que as estações em outra rede vizinha transmitam um tráfego simultâneo.

A segunda diferença é que o alcance de transmissão de estações distintas pode não ser a mesma. Com um fio, o sistema é preparado de modo que todas as estações possam escutar umas às outras. Com a complexidade da propagação por RF, essa situação não acontece para as estações sem fio. Consequentemente, podem surgir situações como o problema do terminal oculto, mencionado anteriormente e ilustrado novamente na Figura 4.23(a). Como nem todas as estações estão dentro do mesmo alcance de rádio, as transmissões que acontecem em uma parte de uma célula podem não ser recebidas em outra parte da mesma célula. Nesse exemplo, a estação *C* está transmitindo para a estação *B*. Se *A* detectar o canal, ela não escutará nada e concluirá incorretamente que agora pode começar a transmitir para *B*. Essa decisão leva a uma colisão.

A situação inversa é o problema do terminal exposto, ilustrado na Figura 4.23(b). Aqui, *B* deseja enviar para *C* e, portanto, escuta o canal. Quando ele detecta uma transmissão, conclui incorretamente que não pode transmitir para *C*, embora *A* de fato possa estar transmitindo para *D* (não mostrado). Essa decisão desperdiça uma oportunidade de transmissão.

Para reduzir ambiguidades sobre qual estação está transmitindo, o 802.11 define a detecção do canal de maneira física e virtual. A detecção física simplesmente verifica o meio para ver se existe um sinal válido. Com a detecção virtual, cada estação mantém um registro lógico de quando o canal está em uso rastreando o **vetor de alocação de rede**, ou **NAV** (**Network Allocation Vector**). Todo quadro transporta um campo NAV que diz quanto tempo levará para concluir a sequência da qual esse quadro faz parte. As estações que escutam esse quadro sabem que o canal estará ocupado pelo período indicado pelo NAV, independentemente se elas podem detectar um sinal físico. Por exemplo, o NAV de um quadro de dados inclui o tempo necessário para enviar uma confirmação. Todas as estações que escutarem o quadro de dados serão adiadas durante o período de confirmação, mesmo que não a escutem.

Um mecanismo RTS/CTS opcional usa o NAV para impedir que os terminais transmitam quadros ao mesmo tempo que os terminais ocultos. Isso aparece na Figura 4.24. Nesse exemplo, *A* deseja enviar para *B*. *C* é uma estação dentro do alcance de *A* (e possivelmente dentro do alcance de *B*, mas isso não importa). *D* é uma estação dentro do alcance de *B*, mas não dentro do alcance de *A*.

O protocolo começa quando *A* decide enviar dados para *B*. *A* começa a transmitir um quadro RTS para *B*, pedindo permissão para lhe enviar (Request To Send) um quadro. Se *B* recebe esse pedido, responde com um quadro CTS, indicando que o canal está liberado para enviar (Clear To Send). Ao receber o CTS, *A* envia seu quadro e inicia um timer de ACK (confirmação). Ao recebimento correto do quadro de dados, a estação *B* responde com um quadro ACK, comple-

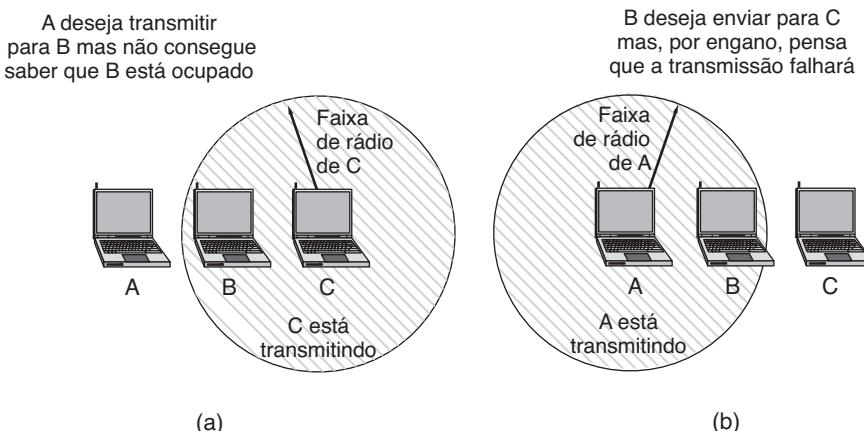



Figura 4.23 | (a) O problema do terminal oculto. (b) O problema do terminal exposto.

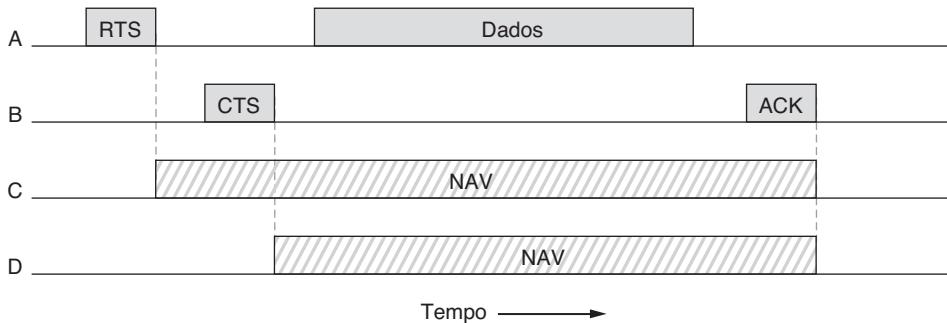



Figura 4.24 | O uso da detecção de canal virtual com o CSMA/CA.

tando a troca. Se o timer de ACK de *A* expirar antes que o ACK retorne a ela, isso é tratado como uma colisão e o protocolo inteiro é realizado novamente, após um backoff.

Agora, vamos considerar essa troca do ponto de vista de *C* e *D*. Como *C* está dentro do alcance de *A*, ela pode receber o quadro RTS. Se receber, essa estação percebe que alguém transmitirá dados em breve. Pela informação fornecida no pedido de RTS, ela pode estimar o tempo que a sequência levará, incluindo o ACK final. Assim, para o bem de todos, a estação C desiste de transmitir algo até que a troca seja concluída. Ela faz isso atualizando seu registro do NAV para indicar que o canal está ocupado, como mostra a Figura 4.24. *D* não escuta o RTS, mas escuta o CTS, de modo que também atualiza seu NAV. Observe que os sinais NAV não são transmitidos; eles são apenas lembretes internos para ficar em silêncio por determinado período.

Entretanto, embora RTS/CTS pareça ser bom na teoria, esse é um daqueles projetos que provaram ter pouco valor na prática. Existem vários motivos pelos quais ele raramente é usado. Ele não ajuda para quadros curtos (que são enviados no lugar do RTS) ou para o PA (que, por definição, todos podem ouvir). Para outras situações, ele só atrasa a operação. RTS/CTS no 802.11 é um pouco diferente daquele do protocolo MACA, que vimos na Seção 4.2, pois qualquer um que escuta o RTS ou o CTS permanece em silêncio por todo o período, para permitir que o ACK seja enviado sem colisão. Por causa disso, a técnica não ajuda com terminais expostos como o MACA fazia, somente com terminais ocultos. Com frequência, existem poucos terminais ocultos, e o CSMA/CA já os ajuda atrasando as estações que não têm êxito na transmissão, qualquer que seja a causa, para aumentar as chances de êxito das transmissões.

O CSMA/CA com detecção física e virtual é o núcleo do protocolo 802.11. Porém, existem outros mecanismos que foram desenvolvidos para acompanhá-lo. Como cada um desses mecanismos foi controlado pelas necessidades da operação real, vamos examiná-los rapidamente.

A primeira necessidade que examinaremos é a confiabilidade. Em contraste com as redes fisicamente conectadas, as redes sem fios são ruidosas e pouco confiáveis, em grande parte em virtude da interferência com outros dispositivos, como os fornos de micro-ondas, que também utilizam as bandas ISM não licenciadas. O uso de confirmações e retransmissões não ajuda muito se a probabilidade de transferir um quadro for pequena em primeiro lugar.

A estratégia principal usada para aumentar as transmissões com êxito é reduzir a taxa de transmissão. Taxas mais baixas usam modulações mais robustas, que mais provavelmente serão recebidas de modo correto para determinada relação sinal-ruído. Se muitos quadros se perderem, uma estação poderá reduzir a taxa. Se os quadros forem entregues com pouca perda, uma estação ocasionalmente poderá testar uma taxa mais alta para ver se ela deve ser usada.

Outra estratégia para melhorar as chances de o quadro atravessar a rede sem prejuízo é enviar quadros mais curtos. Se a probabilidade de ocorrer um erro em qualquer bit é p, então a probabilidade de um quadro de n bits ser recebido de forma inteiramente correta é $(1 - p)^n$. Por exemplo, para $p = 10^{-4}$, a probabilidade de receber um quadro Ethernet completo (12.144 bits) sem erros é menor que 30 por cento. A maioria dos quadros será perdida. Mas, se os quadros tiverem apenas um terço desse tamanho (4.048 bits), dois terços deles serão recebidos corretamente. Agora, a maioria dos quadros passará e menos retransmissões serão necessárias.

Quadros mais curtos podem ser implementados reduzindo o tamanho máximo da mensagem que é aceita a partir da camada de rede. Como alternativa, o 802.11 permite que os quadros sejam divididos em partes menores, chamadas **fragmentos**, cada uma com seu próprio checksum. O tamanho do fragmento não é fixado pelo padrão, mas é um parâmetro que pode ser ajustado pelo PA. Os fragmentos são numerados individualmente e confirmados com o uso de um protocolo do tipo stop-and-wait (isto é, o transmissor não pode enviar o fragmento $k + 1$ enquanto não receber a confirmação do fragmento k). Depois que um canal é 'apoderado', vários fragmentos podem ser enviados em rajada. Eles seguem um após o outro com uma confirmação (e, possivelmente, retransmissões) no intervalo, até que o quadro inteiro tenha sido transmitido com sucesso ou o tempo de transmissão atinja o máximo permitido. O

mecanismo NAV mantém as outras estações inativas apenas até a próxima confirmação, mas outro mecanismo (ver a seguir) é usado para permitir que uma rajada de fragmentos seja enviada sem que outras estações enviem um quadro no meio.

A segunda necessidade que discutiremos é economizar energia. A duração da bateria é sempre um problema nos dispositivos móveis sem fios. O padrão 802.11 dedica atenção à questão do gerenciamento de energia, para que os clientes não a desperdicem quando não têm informações para enviar ou para receber.

O mecanismo básico para economizar energia é calcado em **quadros de baliza**. As balizas são transmissões periódicas do PA (por exemplo, a cada 100 ms). Os quadros de baliza anunciam a presença do PA aos clientes e transportam parâmetros do sistema, como o identificador do PA, a hora, o tempo até a próxima baliza e configurações de segurança.

Os clientes podem definir um bit de gerenciamento de energia nos quadros que eles enviam ao PA, para informar que estão entrando no **modo de economia de energia**. Nesse modo, o cliente pode cochilar e o PA manterá em buffer o tráfego (barrado) voltado para ele. Para verificar o tráfego que chegou, o cliente acorda a cada baliza e verifica um mapa de tráfego enviado como parte do quadro de baliza. Esse mapa diz ao cliente se existe tráfego à espera no buffer. Se houver, o cliente envia uma mensagem de poll ao PA, que em seguida envia o tráfego armazenado. O cliente pode, então, voltar a dormir até que a próxima baliza seja enviada.

Outro mecanismo de economia de energia, chamado **APSD** (**Automatic Power Save Delivery**), também foi acrescentado ao 802.11 em 2005. Com esse novo mecanismo, o PA mantém quadros em buffer e os envia para um cliente logo depois que o cliente envia quadros ao PA. O cliente pode, então, dormir até que tenha mais tráfego para enviar (e receber). Esse mecanismo funciona bem para aplicações como VoIP, que possuem tráfego frequente nos dois sentidos. Por exemplo, um telefone sem fios VoIP poderia usá-lo para enviar e receber quadros a cada 20 ms, com muito mais frequência do que o intervalo de baliza de 100 ms, enquanto chochila nos intervalos.

A terceira e última necessidade que examinaremos é a qualidade de serviço. Quando o tráfego VoIP no exemplo anterior competir com o tráfego peer-to-peer, o VoIP sofrerá. Ele será adiado em virtude da disputa com o tráfego peer-to-peer de alta largura de banda, embora a largura de banda VoIP seja baixa. Esses atrasos provavelmente degradarão as chamadas de voz. Para impedir essa degradação, gostaríamos de permitir que o tráfego de VoIP siga antes do tráfego peer-to-peer, pois tem maior prioridade.

O IEEE 802.11 tem um mecanismo inteligente para fornecer esse tipo de qualidade de serviço que foi apresentado como um conjunto de extensões sob o nome 802.11e em 2005. Ele funciona estendendo o CSMA/CA com intervalos cuidadosamente definidos entre os quadros. Depois que um quadro é enviado, é necessária uma certa quantidade de tempo de inatividade antes que qualquer estação possa enviar um quadro para verificar se o canal não está mais sendo usado. O truque é definir diferentes intervalos para diferentes tipos de quadros.

Cinco intervalos são representados na Figura 4.25. O intervalo entre quadros de dados regulares é chamado de **espaçamento entre quadros DCF**, ou **DIFS** (**DCF InterFrame Spacing**). Qualquer estação pode tentar adquirir o canal para enviar um novo quadro até que o meio tenha ficado ocioso por DIFS. As regras habituais de disputa se aplicam e, se ocorrer uma colisão, o algoritmo de backoff exponencial binário pode ser necessário. O menor intervalo é o **espaçamento curto entre quadros**, ou **SIFS** (**Short InterFrame Spacing**). Ele é usado para permitir que as partes de um único diálogo tenham a chance de transmitir primeiro. Isso inclui a permissão para que o receptor envie um ACK, outras sequências de quadro de controle, como RTS e CTS, ou deixe que o receptor envie uma rajada de fragmentos. O envio do próximo fragmento após esperar apenas o SIFS é o que impede que outra estação entre com um quadro no meio da troca.

Os dois intervalos **AIFS** (**Arbitration InterFrame Space**) mostram exemplos de dois níveis de prioridade. O intervalo curto, $AIFS_1$, é menor que o DIFS, porém maior que o SIFS. Ele pode ser usado pelo PA para mover o tráfego de voz e outro tráfego de alta prioridade para o início da linha. O PA esperará por um intervalo mais curto antes

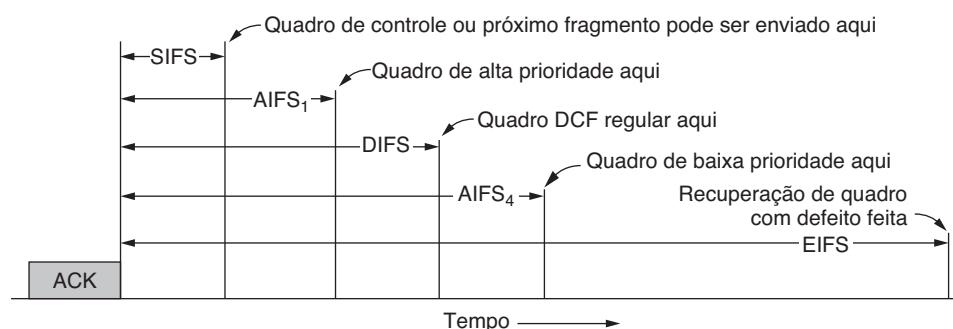



Figura 4.25 | Espaçamento entre quadros no 802.11.

de enviar o tráfego de voz e, assim, o envia antes do tráfego regular. O intervalo longo, $AIFS_4$, é maior que o DIFS. Ele é usado para o tráfego de segundo plano, que pode ser adiado para depois do tráfego regular. O PA esperará por um intervalo maior antes de enviar esse tráfego, dando ao tráfego regular a oportunidade para transmitir primeiro. O mecanismo completo de qualidade de serviço define quatro níveis de prioridade, que têm diferentes parâmetros de backoff, bem como diferentes parâmetros ociosos.

O último intervalo, o **espaçamento estendido entre quadros**, ou **EIFS** (**Extended InterFrame Spacing**), só é usado por uma estação que tenha acabado de receber um quadro defeituoso ou desconhecido, a fim de informar sobre o problema. A ideia é que, como o receptor talvez não tenha nenhum conhecimento do que está acontecendo, ele deve esperar um tempo significativo para evitar interferir em um diálogo em andamento entre duas estações.

Outra parte das extensões de qualidade de serviço é a noção de uma **TXOP**, ou **oportunidade de transmissão** (**transmission oportunity**). O mecanismo de CSMA/CA original permite que as estações enviem um quadro de cada vez. Esse projeto foi bom até o aumento das taxas de transferência. Com o 802.11a/g, uma estação poderia enviar a 6 Mbps e outra estação enviar a 54 Mbps. Cada uma delas passa a enviar um quadro, mas a estação de 6 Mbps leva nove vezes mais tempo (ignorando os overheads fixos) que a estação de 54 Mbps para enviar seu quadro. Essa disparidade tem o efeito colateral de atrasar um transmissor rápido que esteja competindo com um transmissor lento para aproximadamente a taxa do transmissor lento. Por exemplo, novamente ignorando overheads fixos, enviando sozinhos, os transmissores de 6 e 54 Mbps receberão em suas próprias taxas, mas, ao enviar juntos, eles receberão 5,4 Mbps na média. Essa é uma penalidade cruel para o transmissor rápido. Esse problema é conhecido como **anomalia de taxa** (Heusse et al., 2003).

Com oportunidades de transmissão, cada estação recebe uma fatia igual de tempo, não um número igual de quadros. As estações que enviam a uma taxa mais alta que seu tempo receberão um throughput maior. Em nosso exemplo, ao enviar juntos, os transmissores de 6 Mbps e 54 Mbps agora receberão 3 Mbps e 27 Mbps, respectivamente.

4.4.4 802.11: estrutura do quadro

O padrão 802.11 define três classes de quadros em trânsito: dados, controle e gerenciamento. Cada um deles tem um cabeçalho com uma variedade de campos usados na subcamada MAC. Além disso, existem alguns cabeçalhos usados pela camada física, mas eles lidam principalmente com as técnicas de modulação empregadas e, portanto, não os discutiremos aqui.

Veremos como exemplo o formato do quadro de dados mostrado na Figura 4.26. Primeiro vem o campo *Controle de quadro*. Ele próprio tem 11 subcampos. O primeiro desses subcampos denomina-se *Versão do protocolo*, definido como 00. Ele existe para permitir que versões futuras do 802.11 operem ao mesmo tempo na mesma célula. Depois, temos os campos *Tipo* (dados, controle ou gerenciamento) e *Subtipo* (por exemplo, RTS ou CTS). Para um quadro de dados regular (sem qualidade de serviço), eles são definidos como 10 e 0000 em binário. Os bits *Para DS* e *De DS* indicam se o quadro está indo ou vindo da rede conectada aos PAs, o que é chamado sistema de distribuição. O bit *Mais Fragmentos* significa que mais fragmentos virão em seguida. O bit *Repetir* indica uma retransmissão de um quadro enviado anteriormente. O bit *Gerenciamento de energia* indica que o transmissor está entrando no modo de economia de energia. O bit *Mais dados* indica que o transmissor tem quadros adicionais para o receptor. O bit *Quadro protegido* especifica que o corpo do quadro foi criptografado por segurança. Discutiremos rapidamente sobre segurança na próxima seção. Por fim, o bit *Ordem* informa ao receptor que a camada superior espera que a sequência de quadros chegue estritamente em ordem.

O segundo campo do quadro de dados, *Duração*, informa por quanto tempo o quadro e sua confirmação ocuparão o canal, medido em microssegundos. Esse campo está presente em todos os tipos de quadros, incluindo os de controle, e representa a forma como outras estações administram o mecanismo NAV.

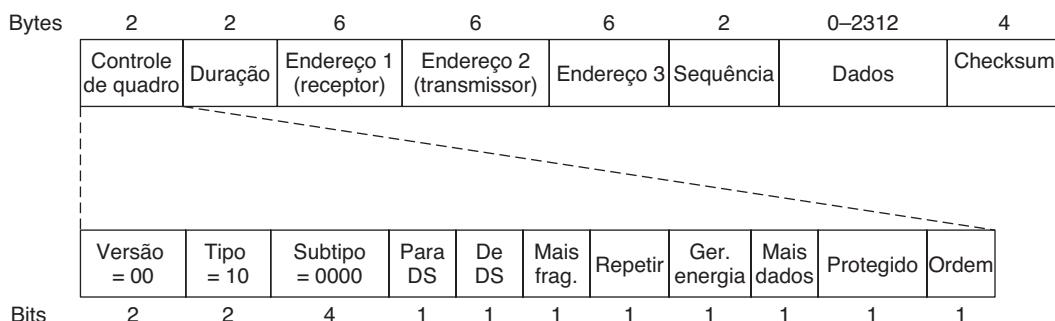



Figura 4.26 | Formato do quadro de dados 802.11.

Em seguida vêm os endereços. Os quadros de dados enviados de e para um PA contêm três endereços, todos em formato padrão IEEE 802. O primeiro endereço é do receptor, e o segundo é do transmissor. É óbvio que eles são necessários, mas para que serve o terceiro endereço? Lembre-se de que o PA é simplesmente um ponto de repasse para os quadros enquanto trafegam entre um cliente e outro ponto na rede, talvez um cliente distante ou um portal para a Internet. O terceiro endereço indica esse ponto distante.

O campo *Sequência* permite que os fragmentos sejam numerados. Dos 16 bits disponíveis, 4 identificam o fragmento e 12 contêm um número que é avançado a cada nova transmissão. O campo *Dados* contém a carga útil de até 2.312 bytes. Os primeiros bytes dessa carga útil estão em um formato conhecido como **controle lógico do enlace**, ou **LLC** (**Logical Link Control**). Essa camada é a tag (a cola) que identifica o protocolo de nível mais alto (por exemplo, IP) ao qual as cargas úteis devem ser passadas. Por último vem o *Checksum do quadro*, que é o mesmo CRC de 32 bits que vimos na Seção 3.2.2 e em outros lugares.

Os quadros de gerenciamento têm um formato semelhante ao dos quadros de dados, mais um formato para a parte de dados que varia com o subtipo (por exemplo, parâmetros nos quadros de baliza). Os quadros de controle são curtos. Como todos os quadros, eles têm os campos de *Controle de quadro*, *Duração* e *Checksum do quadro*. Porém, eles podem ter apenas um endereço e nenhuma parte de dados. A informação mais importante está no campo *Subtipo* (por exemplo, ACK, RTS e CTS).

4.4.5 Serviços

O padrão 802.11 define que cada LAN sem fio compatível deve fornecer os serviços para clientes, pontos de acesso e a rede que os conecta. Esses serviços são agrupados em vários tipos.

O serviço de **associação** é usado pelas estações móveis para conectá-las aos PAs. Em geral, ele é usado imediatamente após uma estação se deslocar dentro do alcance de rádio do PA. Ao chegar, a estação descobre a identidade e os recursos do PA, seja pelos quadros de baliza, seja perguntando diretamente ao PA. Os recursos incluem as taxas de dados admitidas, os arranjos de segurança, requisitos de economia de energia, suporte para qualidade de serviço e outros. A estação envia um pedido para se associar ao PA, o qual pode aceitar ou rejeitar o pedido.

A **reassociação** permite mudar seu PA preferido. Esse recurso é útil para estações móveis que se deslocam de um PA para outro na mesma LAN 802.11 estendida, como um handoff na rede celular. Se for usado corretamente, não haverá perda de dados em consequência do handoff. (Porém, o 802.11, como o padrão Ethernet, é apenas um serviço que faz o melhor possível.) A estação móvel ou o PA também pode se **desassociar**, interrompendo assim o relacionamento. Uma estação deve usar esse serviço antes de se desligar ou sair da rede, mas o PA também pode usá-lo antes de se desativar para manutenção.

As estações também devem se **autenticar** antes que possam enviar quadros pelo PA, mas a autenticação é tratada de diferentes maneiras, dependendo da escolha do esquema de segurança. Se a rede 802.11 estiver 'aberta', qualquer um tem permissão para usá-la. Caso contrário, são necessárias credenciais para autenticar. O esquema recomendado, chamado **WPA2** (**WiFi Protected Access 2**), implementa a segurança conforme a definição no padrão 802.11i. (O WPA original é um esquema intermediário que implementa um subconjunto do 802.11i. Pularemos isso e iremos diretamente para o esquema completo.) Com o WPA2, o PA pode falar com um servidor de autenticação, que tem um banco de dados de nomes de usuários e senhas, para determinar se a estação tem permissão para acessar a rede. Como alternativa, uma chave previamente compartilhada, que é um nome elegante para uma senha de rede, pode ser configurada. Vários quadros são trocados entre a estação e o PA com um desafio e resposta que permite que a estação prove que tem as credenciais corretas. Essa troca acontece após a associação.

O esquema que foi usado antes do WPA é chamado **WEP (Wired Equivalent Privacy)**. Para esse esquema, a autenticação com uma chave previamente compartilhada acontece antes da associação. Contudo, seu uso é desencorajado em decorrência de falhas no projeto que tornam o WEP fácil de burlar. A primeira demonstração prática de que o WEP foi quebrado apareceu quando Adam Stubblefield era um estagiário de verão na AT&T (Stubblefield et al., 2002). Ele foi capaz de codificar e testar um ataque em uma semana, grande parte desse tempo foi gasto para obter permissão da gerência para comprar as placas WiFi necessárias para as experiências. O software para descobrir senhas WEP agora está disponível gratuitamente.

Quando os quadros alcançam o PA, o serviço de **distribuição** determina como roteá-los. Se o destino for local para o PA, os quadros poderão ser enviados diretamente pelo ar. Caso contrário, eles terão de ser encaminhados pela rede fisicamente conectada. O serviço de **integração** trata de qualquer tradução necessária para um quadro ser enviado fora da LAN 802.11, ou para chegar de fora dela. O caso comum aqui é conectar a LAN sem fios à Internet.

A transmissão de dados é o objetivo de tudo isso, e assim o 802.11 oferece um serviço de **entrega de dados**. Esse serviço permite que as estações transmitam e recebam dados usando os protocolos que descrevemos anteriormente no capítulo. Tendo em vista que o 802.11 foi modelado com base no padrão Ethernet e que a transmissão em uma rede Ethernet não oferece a garantia de ser 100 por cento confiável, a transmissão sobre redes 802.11 também não oferece nenhuma garantia de confiabilidade.

As camadas mais altas devem lidar com a detecção e a correção de erros.

A LAN sem fios usa um sinal de broadcast. Para que as informações enviadas por uma LAN sem fios sejam mantidas confidenciais, elas devem ser criptografadas. Esse objetivo é realizado com um serviço de **privacidade** que gerencia os detalhes da criptografia e da descriptografia. O algoritmo de criptografia para WPA2 é baseado no **padrão de criptografia avançado**, ou **AES** (**Advanced Encryption Standard**), um padrão do governo dos Estados Unidos aprovado em 2002. As chaves usadas para criptografia são determinadas durante o procedimento de autenticação.

Para lidar com o tráfego com diferentes prioridades, existe um serviço de **escalonamento de tráfego QOS**. Ele usa os protocolos que descrevemos para dar tratamento preferencial ao tráfego de voz e vídeo em comparação com o melhor tráfego possível e o de segundo plano. Um serviço de acompanhamento também oferece sincronização de timer da camada mais alta. Isso permite que as estações coordenem suas ações, o que pode ser útil para o processamento de mídia.

Finalmente, existem dois serviços que ajudam as estações a gerenciar seu uso do espectro. O serviço de **controle de potência de transmissão** oferece às estações as informações que elas precisam para atender aos limites regulamentares sobre potência de transmissão, que variam de uma região para outra. O serviço de **seleção dinâmica de frequência** dá às estações a informação de que elas precisam para evitar transmitir em frequências na banda de 5 GHz que estão sendo usadas em um radar nas proximidades.

Com esses serviços, o 802.11 oferece um rico conjunto de funcionalidade para conectar à Internet clientes móveis vizinhos. Ele tem sido um grande sucesso, e o padrão repetidamente tem sido alterado para acrescentar mais funcionalidade. Para ter uma ideia de onde o padrão se encontra e para onde está se encaminhando, consulte Hiertz et al. (2010).

4.5 Redes de banda larga sem fios

Passamos muito tempo cuidando de ambientes internos. Agora, vamos sair e ver se há algo interessante acontecendo lá fora em relação a redes, no chamado 'último quilômetro'. Com a privatização do sistema de telefonia em muitos países, os concorrentes que disputam as empresas com frequência têm permissão para oferecer serviços locais de voz e Internet de alta velocidade. Sem dúvida, há uma grande demanda por esses serviços. O problema é que estender cabos de fibra, coaxiais ou mesmo de par trançado Categoria 5 até milhões de residências e escritórios é algo proibitivamente dispendioso. O que uma empresa concorrente deve fazer?

A resposta é a rede sem fio de banda larga. Erguer uma grande antena em uma colina fora da cidade e instalar antenas orientadas nos telhados dos clientes é muito mais fácil e econômico que cavar valas e estender cabos. Desse modo, as empresas de telecomunicações concorrentes têm um grande interesse em fornecer um serviço de comunicação sem fio de vários megabits para voz, Internet, filmes sob demanda etc.

Para estimular o mercado, o IEEE formou um grupo para padronizar uma rede metropolitana sem fio de banda larga. O próximo número disponível no espaço de numeração 802 foi **802.16**, de modo que o padrão recebeu esse número. Informalmente, a tecnologia é chamada **WiMAX** (**Worldwide Interoperability for Microwave Access**). Usaremos os termos 802.16 e WiMAX para indicar a mesma coisa.

O primeiro padrão 802.16 foi aprovado em dezembro de 2001. As primeiras versões ofereciam um circuito terminal sem fios entre pontos fixos, com uma linha de visão de um para outro. Esse projeto logo mudou para tornar o WiMAX uma alternativa mais competitiva ao cabo e DSL para acesso à Internet. Em janeiro de 2003, o 802.16 tinha sido revisado para dar suporte a enlaces fora da linha de visão, usando tecnologia OFDM em frequências entre 2 GHz e 10 GHz. Essa mudança tornou a implantação muito mais fácil, embora as estações ainda fossem locais fixos. O aumento das redes de celular 3G impôs uma ameaça, prometendo altas taxas de dados *e* mobilidade. Em resposta, o 802.16 foi melhorado novamente para permitir mobilidade em velocidades veiculares em dezembro de 2005. O acesso à Internet móvel de banda larga é o alvo do padrão atual, o IEEE 802.16-2009.

Assim como outros padrões 802, o 802.16 foi bastante influenciado pelo modelo OSI, incluindo as (sub)camadas, a terminologia, os primitivos de serviço e outros. Infelizmente, também como o OSI, ele é bastante complicado. De fato, o **WiMAX Fórum** foi criado para definir subconjuntos interoperáveis do padrão para ofertas comerciais. Nas próximas seções, daremos uma breve descrição de alguns dos destaques das formas comuns da interface de ar do 802.16, mas esse tratamento está longe de ser completo e omite muitos detalhes. Para obter informações adicionais sobre WiMAX e a banda larga sem fio em geral, consulte Andrews et al. (2007).

4.5.1 Comparação entre o 802.16 e o 802.11 e 3G

Neste ponto, você poderá estar pensando: por que elaborar um novo padrão? Por que simplesmente não usar 802.11 ou 3G? De fato, o WiMAX combina aspectos do 802.11 e 3G, o que o torna mais semelhante à tecnologia 4G.

Assim como o 802.11, o WiMAX trata da conexão de dispositivos sem fios à Internet em velocidades de megabits/s, em vez de usar cabo ou DSL. Os dispositivos podem ser móveis, ou pelo menos portáteis. O WiMAX não começou acrescentando dados de baixa velocidade no lado de voz das redes de celulares; o 802.16 foi projetado para

transportar pacotes IP pelo ar e conectar-se a uma rede com fios baseada em IP com um mínimo de alvoroço. Os pacotes podem transportar tráfego peer-to-peer, chamadas de VoIP ou streaming de mídia para dar suporte a uma grande faixa de aplicações. Também como o 802.11, ele é baseado na tecnologia OFDM para garantir um bom desempenho apesar das degradações de sinal sem fios, como o enfraquecimento por múltiplos caminhos, e na tecnologia MIMO, para alcançar altos níveis de throughput.

Entretanto, o WiMAX é mais parecido com 3G (e, portanto, diferente do 802.11) em vários aspectos fundamentais. O principal problema técnico é conseguir alta capacidade pelo uso eficiente do espectro, de modo que um grande número de assinantes em uma área de cobertura possa obter um throughput alto. As distâncias típicas são pelo menos dez vezes maiores que para uma rede 802.11. Consequentemente, as estações-base WiMAX são mais poderosas do que os pontos de acesso (PAs) 802.11. Para lidar com sinais mais fracos por distâncias maiores, a estação-base usa mais potência e antenas melhores, e realiza mais processamento para lidar com erros. Para maximizar o throughput, as transmissões são cuidadosamente programadas pela estação-base para cada assinante em particular; o uso do espectro não fica ao acaso com CSMA/CA, o que pode desperdiçar a capacidade com as colisões.

O espectro licenciado é o caso esperado para WiMAX, normalmente em torno de 2,5 GHz nos Estados Unidos. O sistema inteiro é muito mais otimizado do que o 802.11. Essa complexidade compensa, considerando a grande quantidade de dinheiro envolvida para o espectro licenciado. Diferente do 802.11, o resultado é um serviço gerenciado e confiável, com bom suporte para a qualidade de serviço.

Com todos esses recursos, o 802.16 é mais parecido com as redes de celular 4G que agora estão sendo padronizadas sob o nome **LTE** (**Long Term Evolution**). Embora as redes de celular 3G sejam baseadas em CDMA e tenham suporte para voz e dados, as redes de celular 4G serão baseadas em OFDM com MIMO, e visarão aos dados, com a voz sendo apenas uma aplicação. Parece que WiMAX e 4G estão em curso de colisão em termos de tecnologia e aplicações. Talvez essa convergência não seja surpresa, visto que a Internet é a aplicação mais interessante e OFDM e MIMO sejam as tecnologias mais conhecidas para usar o espectro com eficiência.

4.5.2 802.16: arquitetura e pilha de protocolos

A arquitetura 802.16 aparece na Figura 4.27. As estações-base se conectam diretamente à rede de backbone do provedor, que, por sua vez, está conectada à Internet. As estações-base se comunicam com as estações por meio da interface com o ar, sem fios. Existem dois tipos de estações. As estações do assinante permanecem em um local fixo, por exemplo, o acesso à Internet de banda larga para residências. As estações móveis podem receber serviço enquanto estão se movendo, por exemplo, em um carro equipado com WiMAX.

A pilha de protocolos do 802.16 usada para interface com o ar é ilustrada na Figura 4.28. A estrutura geral é semelhante à das outras redes 802, mas tem um número maior de subcamadas. A subcamada inferior lida com a transmissão, e mostramos aqui apenas as opções mais populares do 802.16, o WiMAX fixo e móvel. Existe uma camada física diferente para cada opção. As duas camadas operam no espectro licenciado abaixo de 11 GHz e utilizam OFDM, mas de maneiras diferentes.

Acima da camada física, a camada de enlace de dados consiste em três subcamadas. A inferior lida com privacidade e segurança, que é muito mais crucial para redes públicas externas que para redes privadas internas. Ela cuida da criptografia, da descriptografia e do gerenciamento de chaves.

Em seguida, vem a parte comum da subcamada MAC. É nessa parte que estão localizados os principais protocolos, como o de gerenciamento de canais. De acordo com o modelo, a estação-base controla o sistema. Ela pode programar os canais downlink (isto é, da estação-base para o assinante) de modo muito eficiente, e também desempenha um papel importante no gerenciamento dos canais uplink (isto é, do assinante para a estação-base). Um recurso incomum da subcamada MAC é que, diferentemente do que ocorre

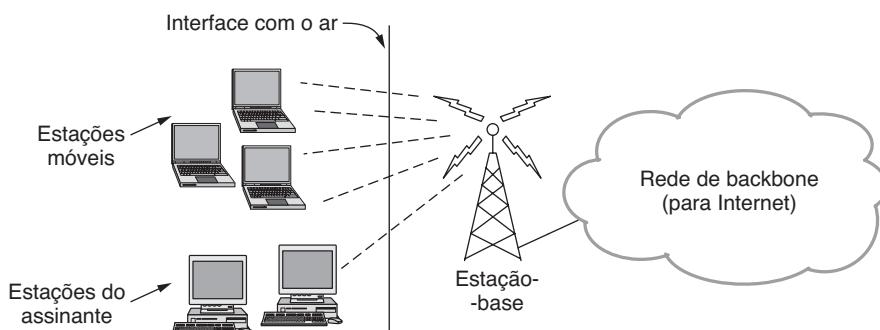



Figura 4.27 | A arquitetura 802.16.

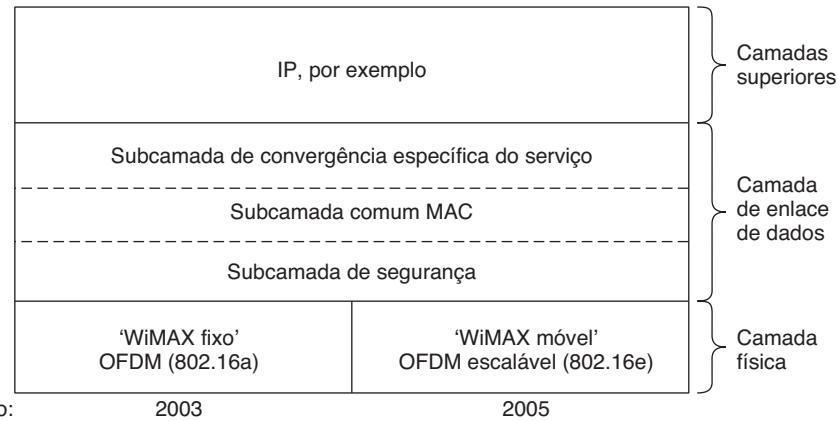



Figura 4.28 | A pilha de protocolos 802.16.

nas outras redes 802, ela é completamente orientada a conexões, a fim de fornecer garantias de qualidade de serviço para a comunicação de telefonia e de multimídia.

A subcamada de convergência de serviços específicos toma o lugar da subcamada de enlace lógico nos outros protocolos 802. Sua função é definir a interface para a camada de rede. Diferentes camadas de convergência são definidas para integrar de modo transparente as diferentes camadas superiores. A escolha importante é o IP, embora o padrão também defina os mapeamentos para protocolos como Ethernet e ATM. Como o IP não é orientado a conexões e a subcamada MAC 802.16 é orientada à conexão, essa camada precisa mapear entre endereços e conexões.

4.5.3 802.16: A CAMADA FÍSICA

A maioria das implementações do WiMAX utiliza o espectro licenciado em torno de 3,5 GHz ou 2,5 GHz. Assim como o 3G, encontrar o espectro disponível é um problema fundamental. Para ajudar, o padrão 802.16 é projetado com flexibilidade. Ele permite a operação de 2 GHz a 11 GHz. Os canais de diferentes tamanhos são aceitos; por exemplo, 3,5 MHz para WiMAX fixo e de 1,25 MHz a 20 MHz para WiMAX móvel.

As transmissões são enviadas por esses canais com OFDM, a técnica que descrevemos na Seção 2.5.3. Em comparação com 802.11, o projeto do OFDM 802.16 é otimizado para obter o máximo do espectro licenciado e das transmissões remotas. O canal é dividido em mais subportadoras com uma duração de símbolos maior, para tolerar maiores degradações do sinal sem fio; parâmetros WiMAX são em torno de 20 vezes superiores aos parâmetros 802.11 comparáveis. Por exemplo, no WiMAX móvel, existem 512 subportadoras para um canal de 5 MHz e o tempo para enviar um símbolo em cada subportadora é de aproximadamente 100 μs.

Os símbolos em cada subportadora são enviados com QPSK, QAM-16 ou QAM-64, esquemas de modulação que descrevemos na Seção 2.5.3. Quando a estação móvel ou de assinante está próxima da estação-base e o sinal recebido tem uma alta relação sinal-ruído (SNR), o QAM-64 pode ser usado para enviar 6 bits por símbolo. Para alcançar estações distantes com um SNR baixo, o QPSK pode ser usado para entregar 2 bits por símbolo. Os dados primeiro são codificados para correção de erro com a codificação convolucional (ou esquemas melhores), que descrevemos na Seção 3.2.1. Essa codificação é comum em canais com ruído, para tolerar alguns erros de bit sem precisar enviar retransmissões. De fato, os métodos de modulação e codificação já devem ser familiares, pois são usados por muitas redes que já estudamos, incluindo 802.11 a cabo e DSL. O resultado disso é que uma estação-base pode dar suporte a até 12,6 Mbps de tráfego downlink e 6,2 Mbps de tráfego uplink por canal de 5 MHz e par de antenas.

Uma coisa que os projetistas do 802.16 não queriam era um certo aspecto do modo como GSM e DAMPS funcionam. Esses dois sistemas utilizam bandas de frequência iguais para tráfego upstream e downstream. Ou seja, eles implicitamente consideram que existe tanto tráfego upstream quanto downstream. Para a voz, o tráfego é simétrico em sua maior parte, mas para o acesso à Internet (e certamente para a navegação Web), normalmente há mais tráfego downstream do que upstream. A razão normalmente é 2:1, 3:1 ou mais:1.

Assim, os projetistas escolheram um esquema flexível para dividir o canal entre estações, chamado **OFDMA** (**Orthogonal Frequency Division Multiple Access**). Com OFDMA, diferentes conjuntos de subportadoras podem ser atribuídos a diferentes estações, de modo que mais de uma estação possa enviar ou receber ao mesmo tempo. Se isso fosse 802.11, todas as subportadoras seriam usadas por uma estação para enviar em determinado momento. A flexibilidade adicional no modo como a largura de banda é atribuída pode aumentar o desempenho, pois determinada subportadora poderia ser atenuada em um receptor em decorrência de efeitos por múltiplos caminhos, mas estar

nítida em outro. As subportadoras podem ser atribuídas às estações que podem usá-las melhor.

Assim como ter tráfego assimétrico, as estações normalmente alternam o enviar com o receber. Esse método é chamado **duplexação por divisão de tempo**, ou **TDD** (**Time Division Duplex**). O método alternativo, em que uma estação envia e recebe ao mesmo tempo (em diferentes frequências de subportadora), é chamado **duplexação por divisão de frequência**, ou **FDD** (**Frequency Division Duplex**). O WiMAX permite os dois métodos, mas o TDD é preferido porque é mais fácil de implementar e mais flexível.

A Figura 4.29 mostra um exemplo da estrutura do quadro que é repetida com o tempo. Ela começa com um preâmbulo para sincronizar todas as estações, seguido por transmissões de downlink da estação-base. Primeiro, a estação-base envia mapas para dizer a todas as estações como as subportadoras de downlink e uplink são atribuídas no quadro. A estação-base controla os mapas, de modo que pode alocar diferentes quantidades de largura de banda a estações de quadro para quadro, dependendo das necessidades de cada estação.

Em seguida, a estação-base envia rajadas de tráfego para diferentes estações do assinante (fixas) e móveis nas subportadoras em períodos apresentados no mapa. As transmissões de downlink terminam com um tempo de espera para as estações passarem de recepção para transmissão. Finalmente, as estações do assinante e móvel enviam suas rajadas de tráfego para a estação-base nas posições de uplink que foram reservadas para elas no mapa. Uma dessas rajadas uplink é reservada para o **ranging,** que é o processo pelo qual novas estações ajustam sua temporização e solicitam largura de banda inicial para se conectarem à estação-base. Como nenhuma conexão é configurada nesse estágio, novas estações apenas transmitem e esperam que não haja colisão.

4.5.4 802.16: O PROTOCOLO DA SUBCAMADA MAC 802.16

A camada de enlace de dados é dividida em três subcamadas, como vimos na Figura 4.27. Tendo em vista que só estudaremos criptografia no Capítulo 8, é difícil explicar agora como funciona a subcamada de segurança. Basta saber que a criptografia é usada para manter secretos todos os dados transmitidos. Apenas a carga útil de cada quadro é criptografada; os cabeçalhos não são. Essa propriedade significa que um espião pode ver quem está se comunicando com quem, mas não consegue saber o que uma pessoa está dizendo à outra.

Se você já conhece algo sobre criptografia, aqui está uma explicação de apenas um parágrafo sobre a subcamada de segurança. Se não souber nada sobre criptografia, é provável que você não considere o próximo parágrafo muito esclarecedor (mas talvez fosse interessante ler esse parágrafo outra vez depois de concluir o Capítulo 8).

No momento em que um assinante se conecta a uma estação-base, ele executa um processo de autenticação mútua com criptografia RSA de chave pública, usando certificados X.509. As cargas úteis propriamente ditas são criptografadas com a utilização de um sistema de chave simétrica, seja ele o AES (Rijndael) ou DES com encadeamento de blocos de cifras. A verificação de integridade emprega o SHA-1. Não foi tão ruim assim, foi?

Agora, vamos examinar a parte comum da subcamada MAC. Essa subcamada é orientada à conexão e ponto a multiponto, o que significa que uma estação-base se comunica com várias estações do assinante. Grande parte desse projeto é emprestada dos modems a cabo, em que um headend a cabo controla as transmissões de vários modems a cabo nas instalações do cliente.

O canal downlink é bastante direto. A estação-base controla as rajadas da camada física que são usadas para enviar informações às diferentes estações do assinante. A subcamada MAC simplesmente empacota seus quadros

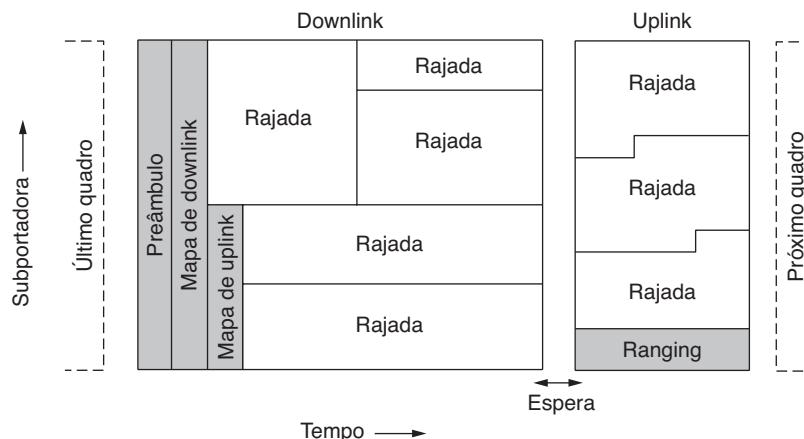



Figura 4.29 | Estrutura de quadro para OFDMA com duplexação por divisão de tempo.

nessa estrutura. Para reduzir o overhead, existem várias opções. Por exemplo, os quadros MAC podem ser enviados individualmente, ou empacotados um após o outro em um grupo.

O canal uplink é mais complicado, pois existem assinantes concorrentes que precisam de acesso a ele. Sua alocação está intimamente relacionada à questão da qualidade de serviço. São definidas quatro classes de serviço, da seguinte forma:

1. Serviço de taxa de bits constante.
2. Serviço de taxa de bits variável em tempo real.
3. Serviço de taxa de bits variável off-line.
4. Serviço de melhor esforço.

Todo serviço no 802.16 é orientado a conexões, e cada conexão recebe uma dessas classes de serviço, determinada quando a conexão é configurada. Essa estrutura é muito diferente da estrutura do 802.11 ou da Ethernet, que não são orientadas a conexões na subcamada MAC.

O serviço de taxa de bits constante se destina à transmissão de voz não compactada, como em um canal T1. Esse serviço precisa enviar uma quantidade de dados predeterminada a intervalos predeterminados. Ele é acomodado dedicando-se certos slots de tempo a cada conexão desse tipo. Uma vez que a largura de banda é alocada, os slots de tempo ficam disponíveis automaticamente, sem a necessidade de solicitar cada um.

O serviço de taxa de bits variável em tempo real se destina a aplicações de multimídia compactada e a outras aplicações de software de tempo real em que a quantidade de largura de banda necessária em cada instante pode variar. Ele é acomodado fazendo-se a estação-base consultar o assinante em intervalos fixos sobre a quantidade de largura de banda necessária em cada momento.

O serviço de taxa de bits variável off-line se destina a transmissões pesadas que não são em tempo real, como as transferências de grandes arquivos. Para esse serviço, a estação-base consulta o assinante com frequência, mas não em intervalos rigidamente prescritos. As conexões com esse serviço também podem usar o serviço do melhor esforço possível, descrito a seguir, para solicitar largura de banda.

Por fim, o serviço de melhor esforço se destina a todos os outros casos. Nenhum polling é feito e o assinante deve disputar a largura de banda com outros assinantes do serviço de melhor esforço. As solicitações de largura de banda são feitas em rajadas marcadas no mapa uplink como disponíveis para disputa. Se uma solicitação for bem-sucedida, seu sucesso será notado no próximo mapa downlink. Se ela não tiver sucesso, os assinantes malsucedidos terão de tentar de novo mais tarde. Para minimizar colisões, é usado o algoritmo de backoff exponencial binário da Ethernet.

4.5.5 802.16: ESTRUTURA DE QUADRO

Todos os quadros MAC começam com um cabeçalho genérico. O cabeçalho é seguido por um carga útil opcional e um checksum (CRC) opcional, como ilustra a Figura 4.30. A carga útil não é necessária em quadros de controle, como os que solicitam slots de canais. O checksum (de forma surpreendente) também é opcional, em razão da correção de erros na camada física e do fato de não ser feita nenhuma tentativa de retransmitir quadros em tempo real. Por que se preocupar com um checksum se não haverá nenhuma tentativa de retransmissão? Mas, se houver um checksum, esse é o CRC padrão IEEE 802 e as confirmações e retransmissões são usadas para confiabilidade.

Apresentaremos a seguir um breve resumo de informações sobre os campos do cabeçalho da Figura 4.30(a). O bit *EC* informa se a carga útil está criptografada. O campo *Tipo* identifica o tipo de quadro, informando principalmente se a compactação e a fragmentação estão presentes. O campo *CI* indica a presença ou a ausência do checksum final. O campo *EK* informa qual das chaves de criptografia está sendo usada (se houver). O campo *Tamanho* fornece o comprimento completo do quadro, incluindo o cabeçalho. O *Identificador de conexão* informa a qual conexão esse quadro pertence. Por fim, o campo *CRC do cabeçalho* é um checksum relativo apenas ao cabeçalho, empregando o polinômio $x^8 + x^2 + x + 1$.

O protocolo 802.16 tem muitos tipos de quadros. Um exemplo de tipo diferente de quadro, usado para solicitar largura de banda, aparece na Figura 4.30(b). Ele começa

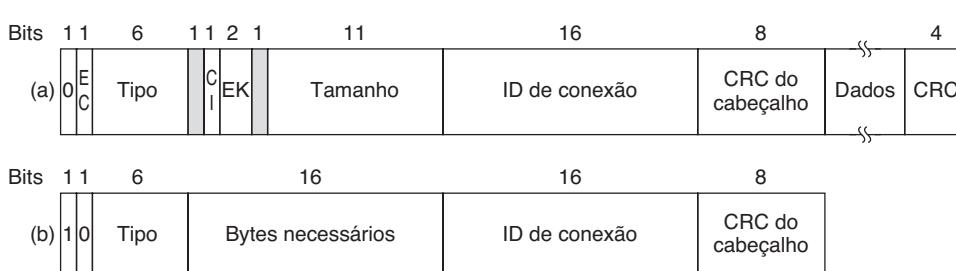



Figura 4.30 | (a) Um quadro genérico. (b) Um quadro de solicitação de largura de banda.

com um bit 1 em vez de um bit 0 e é semelhante ao cabeçalho genérico, exceto pelo fato de que o segundo e o terceiro bytes formam um número de 16 bits que informa a quantidade de largura de banda necessária para transportar o número especificado de bytes. Os quadros de solicitação de largura de banda não transportam uma carga útil ou um CRC para um quadro inteiro.

Poderia ser dito muito mais sobre o padrão 802.16, mas este não é o lugar apropriado. Para obter mais informações, consulte o próprio padrão IEEE 802.16-2009.

4.6 Bluetooth

Em 1994, a empresa L. M. Ericsson ficou interessada em conectar seus telefones móveis a outros dispositivos (por exemplo, laptops) sem cabos. Junto com outras quatro empresas (IBM, Intel, Nokia e Toshiba), ela formou um SIG (Special Interest Group, isto é, um consórcio) com o objetivo de desenvolver um padrão sem fios para interconectar dispositivos de computação e comunicação e ainda acessórios, utilizando rádios sem fios de curto alcance, baixa potência e baixo custo. O projeto foi denominado **Bluetooth**, em homenagem a Harald Blaatand (Bluetooth) II (940-981), um rei viking que unificou (isto é, conquistou) a Dinamarca e a Noruega, também 'sem cabos'.

O Bluetooth 1.0 foi lançado em julho de 1999 e, desde então, o SIG nunca voltou atrás. Todo tipo de dispositivo eletrônico de consumo agora usa Bluetooth, desde telefones móveis e notebooks a headsets, impressoras, teclados, mouse, jogos, relógios, aparelhos de música, unidades de navegação e outros. Os protocolos Bluetooth permitem que esses dispositivos se encontrem e se conectem, um ato chamado **emparelhamento**, e transfiram dados com segurança.

Os protocolos também evoluíram durante a última década. Depois que os protocolos iniciais se estabilizaram, taxas de dados maiores foram acrescentadas ao Bluetooth 2.0 em 2004. Com a versão 3.0, em 2009, o Bluetooth pôde ser usado para o emparelhamento de dispositivo em combinação com o 802.11 para transferência de dados com alto throughput. A versão 4.0, de dezembro de 2009, especificou a operação em baixa potência. Isso será prático para pessoas que não querem trocar a bateria regularmente em todos esses dispositivos por toda a casa. Explicaremos os principais aspectos do Bluetooth a seguir.

4.6.1 Arquitetura do Bluetooth

Vamos começar nosso estudo do sistema Bluetooth com uma avaliação rápida do que ele contém e do que planeja fazer. A unidade básica de um sistema Bluetooth é uma **piconet**, que consiste em um nó mestre e até sete nós escravos ativos, situados dentro de uma distância de dez metros. Podem existir muitas piconets na mesma sala (grande) e elas podem até mesmo ser conectadas por um nó de ponte, como mostra a Figura 4.31. Uma coleção interconectada de piconets é chamada **scatternet**.

Além dos sete nós escravos ativos em uma piconet, pode haver até 255 nós estacionados (inativos) na rede. Esses nós são dispositivos que o mestre comutou para um estado de baixa energia, a fim de reduzir o consumo em sua bateria. No estado estacionário, o dispositivo não pode fazer nada, exceto responder a um sinal de ativação ou de baliza do mestre. Também existem dois níveis intermediários de energia, *hold* e *sniff*, mas esses níveis não serão estudados aqui.

A razão para a estrutura mestre/escravo é que os projetistas pretendiam facilitar a implementação de chips Bluetooth completos por menos de 5 dólares. Em consequência dessa decisão, os escravos são 'não inteligentes', fazendo basicamente apenas o que o mestre determina. Em seu núcleo, uma piconet é um sistema TDM centralizado, no qual o mestre controla o clock e define qual dispositivo vai se comunicar em cada slot de tempo. Toda comunicação é feita entre o mestre e um escravo; não é possível a comunicação direta entre escravos.

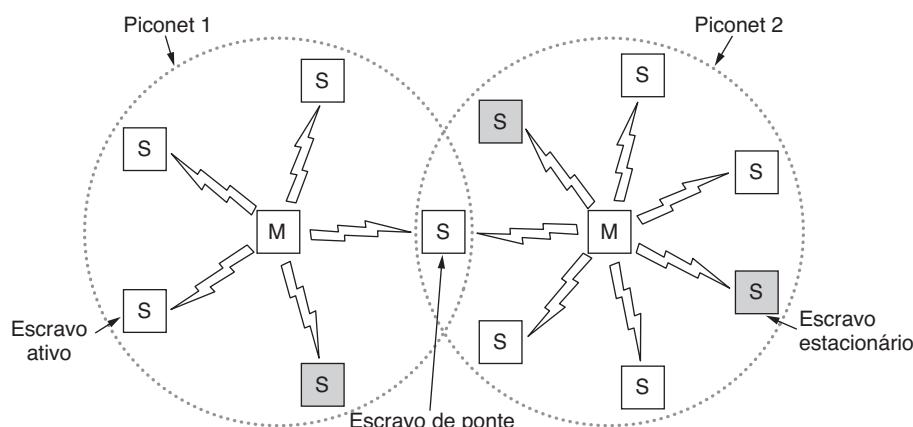



Figura 4.31 | Duas piconets podem ser conectadas para formar uma scatternet.

4.6.2 Aplicações do Bluetooth

A maioria dos protocolos de rede só fornece canais entre entidades que se comunicam, deixando para os projetistas de aplicações a tarefa de descobrir a utilidade desses canais. Por exemplo, o 802.11 não especifica se os usuários devem usar seu notebook para ler correio eletrônico, navegar na Web ou qualquer outra ação. Em contrapartida, a especificação Bluetooth SIG especifica aplicações em particular para que tenham suporte e ofereçam diferentes pilhas de protocolos para cada um. No momento em que este livro foi escrito, havia 25 aplicações específicas, chamadas **perfis**. Infelizmente, essa abordagem aumentou muito a complexidade. Omitiremos a complexidade aqui, mas veremos os perfis rapidamente, para entender de modo mais claro o que o SIG do Bluetooth está tentando realizar.

Seis dos perfis são para diferentes usos de áudio e vídeo. Por exemplo, os perfis de intercomunicação permitem que dois telefones se conectem como 'walkie-talkies'. Os perfis de headset e hands-free oferecem comunicação por voz entre um headset e sua estação-base, pois poderiam ser usados para telefonia hands-free enquanto se dirige um carro. Outros perfis são para streaming de áudio e vídeo com qualidade de estéreo, digamos, de um aparelho de música portátil para fones de ouvido, ou de uma câmera digital para uma TV.

O perfil de dispositivo de interface humana é para conectar teclado e mouse aos computadores. Outros perfis permitem que um telefone móvel ou outro computador receba imagens de uma câmera ou envie imagens para uma impressora. Talvez seja mais interessante um perfil para usar um telefone móvel como um controle remoto para uma TV (habilitada para Bluetooth).

Outros perfis ainda permitem o uso de rede. O perfil de rede pessoal permite que dispositivos Bluetooth formem uma rede ad hoc ou acessem outra rede remotamente, como uma LAN 802.11, por meio de um ponto de acesso. O perfil de rede discada foi realmente a motivação original para o projeto inteiro. Ele permite que um notebook se conecte a um telefone móvel contendo um modem embutido, sem usar fios.

Os perfis para troca de informações da camada mais alta também foram definidos. O perfil de sincronização serve para carregar dados para um telefone móvel quando ele sai de casa e coleta dados dele ao retornar.

Pularemos o restante dos perfis, exceto para mencionar que alguns servem como blocos de montagem sobre os quais os perfis citados são baseados. O perfil de acesso genérico, no qual todos os outros perfis são baseados, oferece um modo de estabelecer e manter enlaces seguros (canais) entre o mestre e os escravos. Os outros perfis genéricos definem os fundamentos da troca de objeto e transporte de áudio e vídeo. Os perfis utilitários são muito usados para funções como emular uma linha serial, o que é especialmente útil para muitas aplicações legadas.

Seria realmente necessário explicar todas essas aplicações em detalhes e fornecer diferentes pilhas de protocolos para cada uma? É provável que não, mas surgiram diversos grupos de trabalho que elaboraram partes distintas do padrão, e cada um se concentrou em seu problema específico e gerou seu próprio perfil. Imagine tudo isso como uma aplicação da lei de Conway. (Na edição de abril de 1968 da revista *Datamation*, Melvin Conway observou que, se designar *n* pessoas para escrever um compilador, você obterá um compilador de *n* passagens ou, de modo mais geral, a estrutura de software reflete a estrutura do grupo que o produziu.) Provavelmente teria sido possível concluir o trabalho com duas pilhas de protocolos em vez de 25, uma para transferência de arquivos e uma para comunicação em tempo real.

4.6.3 A pilha de protocolos do Bluetooth

O padrão Bluetooth tem muitos protocolos agrupados livremente em camadas, como mostra a Figura 4.32. A primeira observação a fazer é que a estrutura não segue o modelo OSI, o modelo TCP/IP, o modelo 802 ou qualquer outro modelo conhecido.

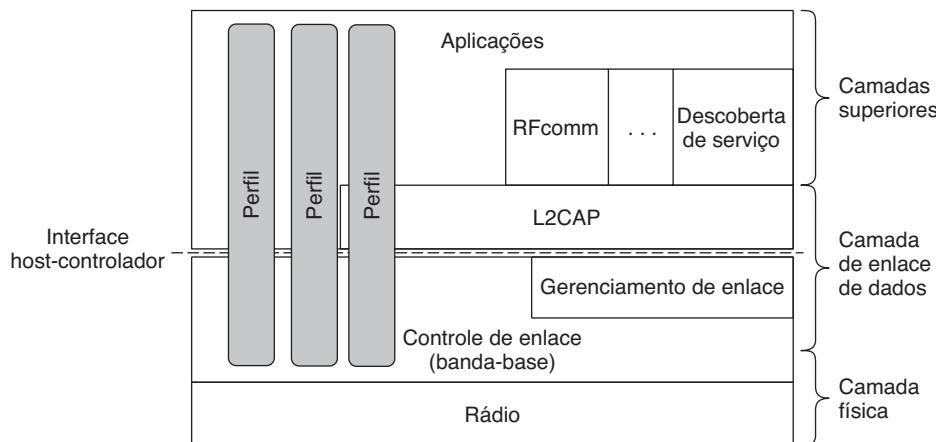



Figura 4.32 | A arquitetura de protocolos do Bluetooth.

A camada inferior é a camada física de rádio, que corresponde muito bem à camada física nos modelos OSI e 802. Ela lida com a transmissão e a modulação de rádio. Muitas das preocupações aqui estão relacionadas ao objetivo de tornar o sistema mais econômico, para que possa vir a ser um item do mercado de massa.

A camada de controle de enlace (ou banda-base) é de certa forma análoga à subcamada MAC, mas também inclui elementos da camada física. Ela lida com a maneira como o mestre controla os slots de tempo e como esses slots estão agrupados em quadros.

Em seguida, temos dois protocolos que usam o protocolo de controle de enlace. O gerenciador de enlaces cuida do estabelecimento de canais lógicos entre dispositivos, incluindo gerenciamento de energia, emparelhamento e criptografia e qualidade de serviço. Ele se encontra abaixo da linha de interface do controle de host. Essa interface é uma conveniência para a implementação: normalmente, os protocolos abaixo da linha serão implementados em um chip Bluetooth, e os protocolos acima dela serão implementados no dispositivo Bluetooth que hospeda o chip.

O protocolo de enlace acima da linha é o **L2CAP** (**Logical Link Control Adaptation Protocol**). Ele enquadra mensagens de tamanho variável e oferece confiabilidade, se necessário. Muitos protocolos utilizam L2CAP, como os dois protocolos utilitários mostrados. O protocolo de descoberta de serviço é usado para localizar serviços dentro da rede. O protocolo RFcomm (comunicação por radiofrequência) simula a porta serial-padrão encontrada nos PCs para a conexão do teclado, mouse e modem, entre outros dispositivos.

A camada superior é onde as aplicações estão localizadas. Os perfis são representados por caixas verticais, pois cada uma delas define uma fatia da pilha de protocolos para determinada finalidade. Perfis específicos, como o de headset, normalmente contêm apenas os protocolos necessários para essa aplicação e nenhum outro. Por exemplo, os perfis podem incluir o L2CAP se tiverem pacotes para enviar, mas o pulam se tiverem apenas um fluxo contínuo de amostras de áudio.

Nas próximas seções, examinaremos a camada de rádio Bluetooth e diversos protocolos de enlace, pois eles correspondem aproximadamente à camada física e à subcamada MAC nas outras pilhas de protocolos que estudamos.

4.6.4 A camada de rádio do Bluetooth

A camada de rádio move os bits do mestre para o escravo ou vice-versa. Ela é um sistema de baixa potência com um alcance de dez metros, operando na banda ISM de 2,4 GHz como o 802.11. A banda está dividida em 79 canais de 1 MHz cada um. Para coexistir com as outras redes usando a banda ISM, é utilizado o espectro de espalhamento por salto de frequência. Pode haver até 1.600 saltos/s pelos slots com um tempo de parada de 625 μs. Todos os nós em uma piconet mudam de frequência simultaneamente, seguindo a temporização de slot e a sequência de salto pseudoaleatória ditada pelo mestre.

Infelizmente, as primeiras versões de Bluetooth e 802.11 interferiram o suficiente para arruinar as transmissões um do outro. Algumas empresas responderam banindo o Bluetooth completamente, mas, por fim, uma solução técnica foi elaborada. A solução é que o Bluetooth adapte sua sequência de saltos para excluir canais em que existam outros sinais de RF. Esse processo reduz a interferência prejudicial. Ele é chamado **salto de frequência adaptativo**.

Três formas de modulação são usadas para enviar bits em um canal. O esquema básico é usar o chaveamento por mudança de frequência para enviar um símbolo de 1 bit a cada microssegundo, dando uma taxa de dados bruta de 1 Mbps. As taxas melhoradas foram introduzidas com a versão 2.0 do Bluetooth. Essas taxas utilizam o chaveamento por deslocamento de fase para enviar 2 ou 3 bits por símbolo, para taxas de dados brutas de 2 ou 3 Mbps. As taxas melhoradas são usadas apenas na parte de dados dos quadros.

4.6.5 As camadas de enlace do Bluetooth

A camada de controle de enlace (ou banda-base) é a estrutura mais próxima de uma subcamada MAC que o Bluetooth tem. Ela transforma o fluxo bruto de bits em quadros e define alguns formatos importantes. Em sua forma mais simples, o mestre em cada piconet define uma série de slots de tempo de 625 μs, com as transmissões do mestre começando nos slots pares e as transmissões dos escravos começando nos slots ímpares. Esse esquema é a tradicional multiplexação por divisão de tempo, em que o mestre fica com metade dos slots e os escravos compartilham a outra metade. Os quadros podem ter 1, 3 ou 5 slots de duração. Cada quadro tem um overhead de 126 bits para um código de acesso e cabeçalho, mais um tempo de ajuste de 250-260 μs por salto, para permitir que os circuitos de rádio se estabilizem. A carga útil do quadro pode ser criptografada por confidencialidade com uma chave escolhida quando o mestre e o escravo se conectam. Os saltos só acontecem entre os quadros, e não durante um quadro. O resultado é que um quadro de 5 slots é muito mais eficiente do que um quadro de 1 slot, pois o overhead é constante, porém mais dados são enviados.

O protocolo gerenciador de enlace estabelece canais lógicos, chamados **enlaces**, para transportar quadros entre um dispositivo mestre e um escravo que descobriram um ao outro. Um procedimento de emparelhamento é seguido para garantir que os dois dispositivos tenham permissão para se comunicar antes que o enlace seja usado. O antigo método de emparelhamento é que os dois dispositivos sejam configurados com o mesmo número de identificação pessoal ou PIN (Personal Identification Number) de quatro dígitos. O PIN correspondente é o modo como cada disposi-

tivo sabe que está se conectando ao dispositivo remoto correto. Porém, usuários e dispositivos sem criatividade usam PINs padrão, como "0000" e "1234", significando que, na prática, esse método fornece muito pouca segurança.

O novo método de **emparelhamento simples seguro** permite que os usuários confirmem se os dois dispositivos estão exibindo a mesma passkey, ou que observem a passkey em um dispositivo e a insiram no segundo dispositivo. Esse método é mais seguro, pois os usuários não precisam escolher ou definir um PIN. Eles simplesmente confirmam uma passkey mais longa, gerada pelo dispositivo. Naturalmente, ela não pode ser usada em alguns dispositivos com entrada/saída limitada, como um headset portátil.

Quando o emparelhamento está concluído, o protocolo do gerenciador de enlace estabelece os enlaces. Existem dois tipos principais de enlaces para transportar dados do usuário. O primeiro é o **enlace síncrono orientado a conexões**, ou **SCO** (**Synchronous Connection Oriented**). Ele é usado para dados em tempo real, como conexões de telefone. Esse tipo de enlace aloca um slot fixo em cada sentido. Um escravo pode ter até três enlaces SCO com seu mestre. Cada enlace SCO pode transmitir um canal de áudio PCM de 64.000 bps. Em virtude da natureza crítica no tempo dos enlaces SCO, os quadros enviados por eles nunca são retransmitidos. Em vez disso, para aumentar a confiabilidade, pode-se usar a correção de erro direta.

O outro tipo é o enlace **assíncrono não orientado a conexões**, ou **ACL** (**Asynchronous ConnectionLess**). Esse tipo de enlace é usado para dados de comutação de pacotes, disponíveis em intervalos irregulares. O tráfego ACL é entregue com base no melhor serviço possível. Nenhuma garantia é oferecida. Os quadros podem se perder e podem precisar ser retransmitidos. Um escravo só pode ter um enlace ACL com seu mestre.

Os dados enviados por enlaces ACL vêm da camada L2CAP. Essa camada tem quatro funções principais. Primeiro, ela aceita pacotes de até 64 KB das camadas superiores e os divide em quadros para transmissão. Na extremidade distante, os quadros são montados novamente em pacotes. Em segundo lugar, ela lida com a multiplexação e a demultiplexação de várias origens de pacotes. Quando um pacote é montado novamente, a L2CAP determina a qual protocolo da camada superior ele será entregue; por exemplo, RFcomm ou descoberta de serviço. Terceiro, a camada L2CAP lida com controle de erro e retransmissão. Ela detecta os erros e retransmite os pacotes que não foram confirmados. Finalmente, a L2CAP impõe requisitos de qualidade de serviço entre enlaces múltiplos.

4.6.6 A estrutura de quadro do Bluetooth

Há vários formatos de quadros no Bluetooth; o mais importante é apresentado de duas formas na Figura 4.33. Ele começa com um código de acesso que normalmente identifica o mestre, para que os escravos situados dentro do alcance de rádio de dois mestres possam conhecer o destino de cada tráfego. Em seguida, há um cabeçalho de 54 bits contendo campos típicos da subcamada MAC. Se o quadro for enviado na taxa de transferência básica, o campo de dados vem em seguida. Ele tem até 2.744 bits para uma transmissão de cinco slots. Para um único slot de tempo, o formato é o mesmo, exceto pelo fato de o campo de dados ter 240 bits.

Se o quadro for enviado na taxa melhorada, a parte de dados pode ter até duas ou três vezes a quantidade de bits, pois cada símbolo transporta 2 ou 3 bits em vez de 1. Esses dados são precedidos por um campo de espera e um padrão de sincronismo que é usado para mudar para uma taxa de dados mais rápida. Ou seja, o código de acesso e o cabeçalho são transportados na taxa básica e somente a parte dos dados é transportada na taxa mais rápida. Os quadros na taxa melhorada terminam com um final mais curto.

Vamos examinar rapidamente o cabeçalho. O campo *Endereço* identifica qual dos oito dispositivos ativos é o destino do quadro. O campo *Tipo* identifica o tipo de quadro (ACL, SCO, polling ou nulo), o tipo de correção de erros usado no campo de dados, e de quantos slots é a duração do quadro. O bit *Fluxo* é definido por um escravo quando

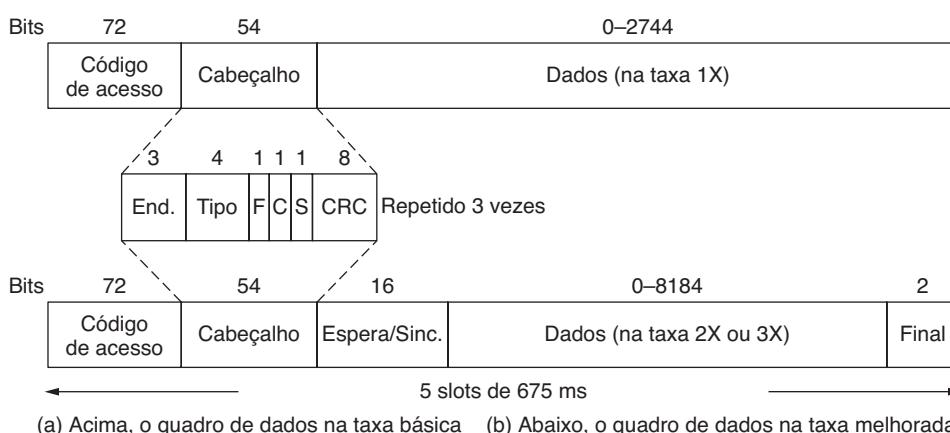



Figura 4.33 | Um quadro de dados típico do Bluetooth nas taxas de dados (a) básica e (b) avançada.

seu buffer está cheio e não pode receber mais dados. Esse bit habilita uma forma primitiva de controle de fluxo. O bit *Confirmação* é usado para transportar uma mensagem ACK em um quadro. O bit *Sequência* é usado para numerar os quadros, a fim de detectar retransmissões. O protocolo é stop-and-wait e, assim, 1 bit é suficiente. Em seguida, temos o cabeçalho de 8 bits *CRC*. O cabeçalho de 18 bits inteiro é repetido três vezes para formar o cabeçalho de 54 bits mostrado na Figura 4.33. No lado receptor, um circuito simples examina todas as três cópias de cada bit. Se todas as três forem iguais, o bit será aceito. Caso contrário, vence a opinião da maioria. Desse modo, 54 bits de capacidade de transmissão são usados para enviar 10 bits de cabeçalho. A razão para isso é que, para transmitir dados de maneira confiável em um ambiente ruidoso usando dispositivos de baixo custo e de baixa potência (2,5 mW), com pouca capacidade de computação, é necessária uma grande redundância.

São usados vários formatos para o campo de dados de quadros ACL e SCO. Entretanto, os quadros SCO são mais simples: o campo de dados tem sempre 240 bits. São definidas três variantes, permitindo 80, 160 ou 240 bits de carga útil real, sendo os bits restantes usados para a correção de erros. Na versão mais confiável (carga útil de 80 bits), o conteúdo é simplesmente repetido três vezes, da mesma forma que o cabeçalho.

Podemos calcular a capacidade com esse quadro da seguinte forma. Tendo em vista que o escravo só pode usar os slots ímpares, ele recebe 800 slots/s, da mesma maneira que o mestre. Com uma carga útil de 80 bits, a capacidade de canal do escravo é de 64.000 bps, assim como a capacidade de canal do mestre. Essa capacidade é exatamente o bastante para um único canal de voz PCM full-duplex (e esse é o motivo de ter sido escolhida uma taxa de saltos de 1.600 saltos/s). Ou seja, apesar de uma largura de banda bruta de 1 Mbps, um único canal de voz satura completamente a piconet. A eficiência de 13 por cento é o resultado de gastar 41 por cento da capacidade no tempo de preparação, 20 por cento com cabeçalhos e 26 por cento na codificação de repetição. Essa deficiência destaca o valor de taxas melhoradas e quadros com mais de um único slot.

Existe muito mais a ser dito sobre o Bluetooth, mas não há mais espaço para isso aqui. Se desejar mais informações, a especificação Bluetooth 4.0 contém todos os detalhes.

4.7 RFID

Vimos os projetos MAC de LANs até MANs e também PANs. Como um último exemplo, estudaremos uma categoria de dispositivos sem fios de classe inferior, que as pessoas podem não reconhecer como formando uma rede de comunicações: as etiquetas e leitoras de **identificação de radiofrequência**, ou **RFID** (**Radio Frequency IDentification**), que descrevemos na Seção 1.5.4.

A tecnologia RFID tem muitas formas, usadas em smartcards, implantes em animais, passaportes, livros de biblioteca e outros. A forma que veremos foi desenvolvida na busca por um **código eletrônico de produto**, ou **EPC** (**Electronic Product Code**), que foi iniciada com o Auto-ID Center no Massachusetts Institute of Technology em 1999. Um EPC é um substituto para o código de barras, que pode transportar uma quantidade maior de informações e é legível eletronicamente por distâncias de até 10 m, mesmo quando não está visível. Essa é uma tecnologia diferente, por exemplo, da RFID usada em passaportes, que deve ser colocada bem perto de uma leitora para realizar uma transação. A capacidade de comunicar a uma certa distância torna os EPCs mais relevantes aos nossos estudos.

A EPCglobal foi formada em 2003 para comercializar a tecnologia RFID desenvolvida pelo Auto-ID Center. O esforço recebeu um impulso em 2005 quando o Walmart exigiu que seus cem maiores fornecedores rotulassem todas as entregas com etiquetas de RFID. A implantação generalizada tem sido prejudicada pela dificuldade de competir com códigos de barra impressos e baratos, porém novos usos, como em carteiras de habilitação, estão ganhando popularidade. Vamos descrever a segunda geração dessa tecnologia, que é chamada informalmente de **EPC Gen 2** (EPCglobal, 2008).

4.7.1 Arquitetura EPC Gen 2

A arquitetura de uma rede de RFID EPC Gen 2 aparece na Figura 4.34. Ela tem dois componentes-chave: etiquetas e leitoras. As etiquetas de RFID são dispositivos pequenos e baratos, que possuem um identificador EPC exclusivo de 96 bits e uma pequena quantidade de memória que pode ser lida e escrita pela leitora de RFID. A memória pode ser usada para registrar o histórico de local de um item, por exemplo, enquanto ele se movimenta na cadeia de suprimentos.

Com frequência, as etiquetas se parecem com adesivos que podem ser afixados, por exemplo, em calças jeans nas prateleiras de uma loja. A maior parte da etiqueta é ocupada por uma antena impressa nela. Um pequeno ponto no meio é o circuito integrado de RFID. Como alternativa, as

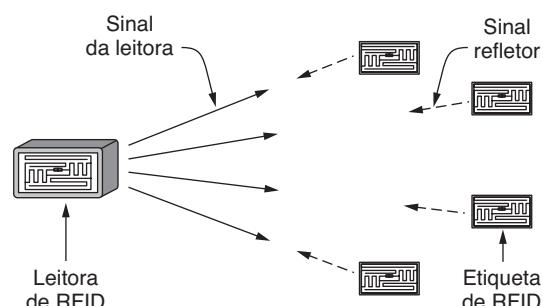



Figura 4.34 | Arquitetura RFID.

etiquetas de RFID podem ser integradas a um objeto, como uma carteira de habilitação. Nos dois casos, as etiquetas não possuem baterias e devem colher energia das transmissões de rádio de uma leitora de RFID nas proximidades para poder funcionar. Esse tipo de etiqueta é chamado etiqueta de "Classe 1", para distingui-la das etiquetas mais capazes, que possuem baterias.

As leitoras são a inteligência no sistema, semelhante às estações-base e aos pontos de acesso nas redes de celular e WiFi. As leitoras são muito mais poderosas que as etiquetas. Elas possuem suas próprias fontes de energia, normalmente têm várias antenas e definem quando as etiquetas enviam e recebem mensagens. Como normalmente haverá muitas etiquetas dentro do alcance de leitura, as leitoras precisam resolver o problema de acesso múltiplo. Também pode haver várias leitoras disputando entre si na mesma área.

A tarefa principal da leitora é fazer o inventário das etiquetas nas vizinhanças, ou seja, descobrir os identificadores das etiquetas vizinhas. O inventário é realizado com o protocolo da camada física e o protocolo de identificação de etiqueta que explicaremos em resumo nas próximas seções.

4.7.2 Camada física EPC Gen 2

A camada física define como os bits são enviados entre a leitora de RFID e as etiquetas. Grande parte dela usa métodos para enviar sinais sem fio que vimos anteriormente. Nos Estados Unidos, as transmissões são enviadas na banda ISM não licenciada de 902-928 MHz. Visto que essa banda cai na faixa do UHF (Ultra High Frequency), as etiquetas são conhecidas como etiquetas UHF RFID. A leitora realiza um salto de frequência pelo menos a cada 400 ms, para espalhar seu sinal pelo canal, para limitar a interferência e cumprir os requisitos regulamentares. A leitora e as etiquetas usam as formas de modulação ASK (Amplitude Shift Keying), que descrevemos na Seção 2.5.2, para codificar os bits. Elas se alternam para enviar bits, de modo que o enlace é half-duplex.

Existem duas diferenças principais em relação as outras camadas físicas que estudamos. A primeira é que a leitora está sempre transmitindo um sinal, independentemente de ser a leitora ou a etiqueta a parte que está se comunicando. Naturalmente, a leitora transmite um sinal para enviar bits às etiquetas. Para as etiquetas enviarem bits para a leitora, esta transmite um sinal de portadora fixa, que não transporta bits. As etiquetas apanham esse sinal para obter a energia de que precisam para funcionar; caso contrário, uma etiqueta não poderia transmitir em primeiro lugar. Para enviar dados, uma etiqueta muda se estiver refletindo o sinal da leitora, como um sinal de radar, batendo em um alvo ou absorvendo-o.

Esse método é chamado **refletor**. Ele é diferente de todas as outras situações sem fio que vimos até aqui, pois transmissor e receptor nunca transmitem ao mesmo tempo. O refletor é um modo de baixa energia para a etiqueta criar um sinal fraco por conta própria, que aparece na leitora. Para que a leitora decodifique o sinal que chega, ela precisa filtrar o sinal de saída que ela mesma está transmitindo. Como o sinal da etiqueta é fraco, elas só podem enviar bits para a leitora em uma taxa baixa, e as etiquetas não podem receber ou mesmo detectar transmissões de outras etiquetas.

A segunda diferença é que são usadas formas de modulação muito simples, para que possam ser implementadas em uma etiqueta que funciona com pouquíssima energia e custa apenas alguns centavos para ser fabricada. Para enviar dados para as etiquetas, a leitora usa dois níveis de amplitude. Os bits são determinados como 0 ou 1, dependendo do tempo que a leitora espera antes de um período de baixa potência. A etiqueta mede o tempo entre os períodos de baixa potência e compara esse tempo com uma referência medida durante um preâmbulo. Como vemos na Figura 4.35, os 1s são maiores que os 0s.

As respostas da etiqueta consistem nesta alternando seu estado refletor em intervalos fixos para criar uma série de pulsos no sinal. Qualquer ponto entre um e oito períodos de pulso pode ser usado para codificar cada 0 ou 1, dependendo da necessidade de confiabilidade. Os 1s têm menos transições que os 0s, como mostramos com um exemplo da codificação no período de dois pulsos da Figura 4.35.

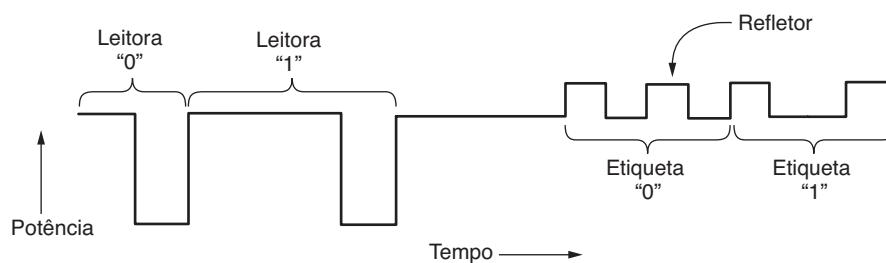



Figura 4.35 | Sinais refletores de leitora e etiqueta.

4.7.3 Camada de identificação de etiqueta EPC Gen 2

Para fazer o inventário de etiquetas vizinhas, a leitora precisa receber uma mensagem de cada etiqueta, oferecendo o identificador para ela. Essa situação é um problema de acesso múltiplo para o qual o número de etiquetas é desconhecido no caso geral. A leitora poderia enviar uma consulta por broadcast para pedir a todas as etiquetas que enviassem seus identificadores. Porém, as etiquetas que respondessem imediatamente colidiriam da mesma forma que as estações em uma rede Ethernet clássica.

Vimos muitas maneiras de enfrentar o problema do acesso múltiplo neste capítulo. O protocolo mais próximo para a situação atual, em que as etiquetas não podem escutar as transmissões umas das outras, é o slotted ALOHA (ou ALOHA segmentado), um dos protocolos mais antigos que estudamos. Esse protocolo é adaptado para uso na RFID Gen 2.

A sequência de mensagens usada para identificar uma etiqueta aparece na Figura 4.36. No primeiro slot (slot 0), a leitora envia uma mensagem *Query* para iniciar o processo. Cada mensagem *QRepeat* avança para o slot seguinte. A leitora também diz às etiquetas o intervalo de slots sobre o qual as transmissões se tornam aleatórias. O uso de um intervalo é necessário porque a leitora sincroniza as etiquetas quando inicia o processo; diferentemente das estações em uma rede Ethernet, as etiquetas não acordam com uma mensagem em um tempo escolhido.

As etiquetas apanham um slot aleatório para responder. Na Figura 4.36, a etiqueta responde no slot 2. Porém, as etiquetas não enviam seus identificadores quando respondem inicialmente. Em vez disso, uma etiqueta envia um número aleatório curto de 16 bits em uma mensagem *RN16*. Se não houver colisão, a leitora recebe essa mensagem e envia uma mensagem *ACK* própria. Nesse estágio, a etiqueta terá adquirido o slot e envia seu identificador de EPC.

Figura 4.36 | Exemplo de troca de mensagem para identificar uma etiqueta.

O motivo para essa troca é que os identificadores EPC são longos, de modo que colisões nessas mensagens seriam dispendiosas. Em vez disso, uma troca curta é usada para testar se a etiqueta pode usar o slot com segurança para enviar seu identificador. Uma vez que seu identificador tiver sido transmitido com êxito, a etiqueta temporariamente deixa de responder a novas mensagens *Query*, para que todas as etiquetas restantes possam ser identificadas.

Um problema importante é que a leitora deve ajustar o número de slots para evitar colisões, mas sem usar muitos slots, para que o desempenho não seja prejudicado. Esse ajuste é semelhante ao backoff exponencial binário da Ethernet. Se a leitora encontrar muitos slots sem respostas ou muitos slots com colisões, ela pode enviar uma mensagem *QAdjust* para diminuir ou aumentar o intervalo dos slots sobre o qual as etiquetas estão respondendo.

A leitora de RFID pode realizar outras operações sobre as etiquetas. Por exemplo, pode selecionar um subconjunto de etiquetas antes de realizar o inventário, permitindo-lhe colher respostas, digamos, de jeans etiquetados, mas não de camisas etiquetadas. A leitora também pode escrever dados nas etiquetas à medida que elas são identificadas. Esse recurso poderia ser usado para registrar o ponto de venda ou outra informação relevante.

4.7.4 Formatos de mensagem de identificação de etiqueta

O formato da mensagem *Query* aparece na Figura 4.37 como um exemplo de mensagem da leitora para a etiqueta. A mensagem é compacta porque as taxas de downlink são limitadas, de 27 kbps até 128 kbps. O campo *Comando* transporta o código 1000 para identificar a mensagem como uma *Query*.

Os próximos flags, *DR*, *M* e *TR*, determinam os parâmetros da camada física para transmissões da leitora e respostas da etiqueta. Por exemplo, a taxa de resposta pode ser definida entre 5 kbps e 640 kbps. Pularemos os detalhes desses flags.

Em seguida vêm três campos, *Sel*, *Sessão* e *Destino*, que selecionam as etiquetas para responder. Assim como as leitoras precisam ser capazes de selecionar um subconjunto de identificadores, as etiquetas precisam acompanhar até quatro sessões simultâneas e se elas foram identificadas nessas sessões. Desse modo, diversas leitoras podem operar nas áreas de cobertura sobrepostas usando diferentes sessões.

Em seguida vem o parâmetro mais importante para esse comando, *Q*. Esse campo identifica o intervalo de slots sobre o qual as etiquetas responderão, de 0 a $2^Q - 1$. Finalmente, existe um CRC para proteger os campos de mensa-

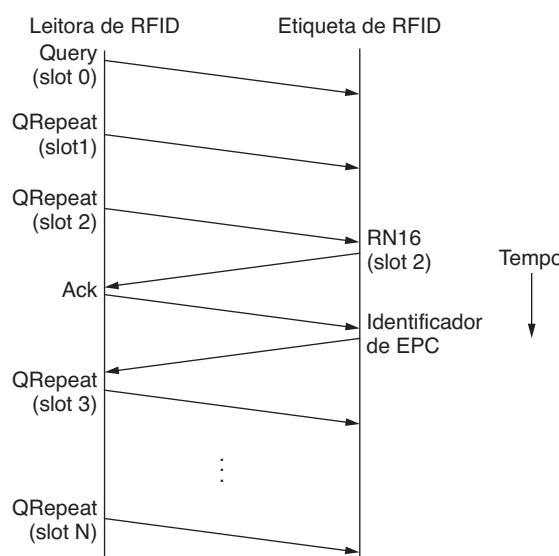
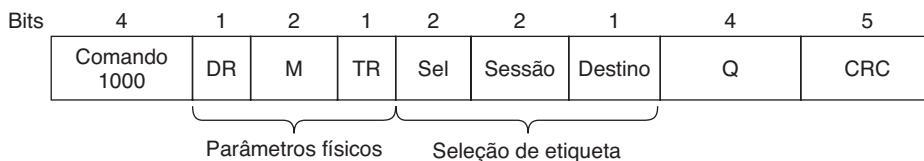



Figura 4.37 | Formato da mensagem Query.

gem. Em 5 bits, ele é mais curto que a maioria dos CRCs que vimos até aqui, mas a mensagem *Query* é muito mais curta do que a maioria dos pacotes também.

As mensagens da etiqueta à leitora são mais simples. Como a leitora está no controle, ela sabe qual mensagem esperar em resposta a cada uma de suas transmissões. As respostas da etiqueta simplesmente transportam dados, como o identificador de EPC.

Originalmente, as etiquetas eram apenas para fins de identificação. Contudo, elas cresceram com o tempo e agora são semelhantes a computadores muito pequenos. Algumas etiquetas de pesquisa têm sensores e são capazes de executar pequenos programas para colher e processar dados (Sample et al., 2008). Uma visão para essa tecnologia é a 'Internet de coisas', que conecta objetos do mundo físico à Internet (Welbourne et al., 2009; Gershenfeld et al., 2004).

4.8 Comutação na camada de enlace de dados

Muitas empresas têm diversas LANs e desejam conectá-las. Não seria conveniente se pudéssemos apenas juntar as LANs para criar uma LAN maior? De fato, podemos fazer isso quando as conexões são feitas com dispositivos chamados **bridges**. Os switches Ethernet que descrevemos na Seção 4.3.4 são um nome moderno para as bridges; eles oferecem funcionalidade que vai além da Ethernet clássica e de hubs Ethernet, facilitando a junção de várias LANs em uma rede maior e mais rápida. Usaremos os termos 'bridge' e 'switch' para indicar a mesma coisa.

As bridges operam na camada de enlace de dados, de modo que examinam os endereços nessa camada para encaminhar quadros. Tendo em vista que elas não têm de examinar o campo de carga útil dos quadros que encaminham, as bridges podem tratar dos pacotes IP ou de quaisquer outros tipos de pacotes, como os pacotes AppleTalk. Em contrapartida, os *roteadores* examinam os endereços em pacotes e efetuam o roteamento com base nesses endereços, de modo que eles só trabalham com os protocolos para os quais foram projetados para lidar.

Nesta seção, examinaremos como as bridges funcionam e como são usadas para juntar várias LANs físicas em uma única LAN lógica. Também veremos como realizar o inverso e tratar uma LAN física como múltiplas LANs lógicas, chamadas **VLANs** (**Virtual LANs**). As duas tecnologias oferecem flexibilidade útil para o gerenciamento de redes. Para ver um estudo abrangente sobre bridges, switches e tópicos relacionados, consulte Seifert e Edwards (2008) e Perlman (2000).

4.8.1 Usos de bridges

Antes de iniciarmos o estudo da tecnologia de bridges, vale a pena examinar algumas situações comuns em que elas são usadas. Mencionaremos três razões pelas quais uma única organização pode ter várias LANs.

Primeiro, muitas universidades e departamentos de empresas têm suas próprias LANs, principalmente para conectar seus computadores pessoais, servidores e dispositivos como impressoras. Como os objetivos dos diversos departamentos são diferentes, muitos deles escolhem LANs distintas, sem se importar com o que outros departamentos estão fazendo. Mais cedo ou mais tarde, surge a necessidade de interação; por isso as bridges são necessárias. Nesse exemplo, a existência de diversas LANs deve-se à autonomia de seus proprietários.

Segundo, a organização pode estar geograficamente dispersa em vários edifícios separados por distâncias consideráveis. Talvez seja mais econômico ter LANs separadas em cada edifício e conectá-las com bridges e enlaces de fibra óptica por longa distância que estender todos os cabos até um único switch central. Mesmo que estender os cabos fosse fácil de fazer, existem limites para seu tamanho (por exemplo, 200 m para a gigabit Ethernet com par trançado). A rede não funcionaria para cabos maiores em virtude de atenuação excessiva do sinal ou pelo atraso de ida e volta. A única solução é partir a LAN e instalar bridges para juntar as partes, aumentando a distância física total que pode ser coberta.

Terceiro, talvez seja necessário dividir aquilo que logicamente é uma única LAN em LANs separadas (conectadas por bridges), a fim de acomodar a carga. Por exemplo, em muitas universidades grandes, há milhares de estações de trabalho disponíveis para as necessidades de computação dos funcionários e dos alunos. As empresas também podem ter milhares de funcionários. A escala desse sistema impede que se coloquem todas as estações de trabalho em uma única LAN — existem mais computadores do que portas em qualquer hub Ethernet, e mais estações do que é permitido em uma única Ethernet clássica.

Mesmo que fosse possível conectar todas as estações de trabalho com fios, colocar mais estações em um hub Ethernet ou na Ethernet clássica não aumentaria a capacidade. Todas as estações compartilham a mesma quantidade fixa de largura de banda. Quanto mais estações houver, menor a largura de banda média por estação.

Entretanto, duas LANs separadas têm o dobro da capacidade de uma única LAN. As bridges permitem que as LANs sejam reunidas enquanto mantêm essa capacidade. O importante é não enviar tráfego para portas onde ele não seja necessário, para que cada LAN possa trabalhar em velocidade plena. Esse comportamento também aumenta a confiabilidade, pois, em uma única LAN, um nó com defeito que continua enviando um fluxo contínuo de lixo pode travar a LAN inteira. Decidindo o que encaminhar e o que não encaminhar, as bridges atuam como portas de incêndio em um prédio, impedindo que um único nó descontrolado trave o sistema inteiro.

Para tornar esses benefícios facilmente disponíveis, o ideal é que as bridges sejam completamente transparentes. Deverá ser possível sair e comprar bridges, conectar os cabos da LAN nas bridges e tudo funcionar perfeitamente, instantaneamente. Não deve ser preciso fazer mudanças de hardware ou de software, nem configuração de endereço de switches, nem baixar tabelas de roteamento ou de parâmetros, nada mesmo. Basta conectar os cabos e sair. Além disso, a operação das LANs existentes não deverá ser afetada de forma alguma pelas bridges. Em relação às estações, não deverá haver diferença observável por estarem ou não fazendo parte de uma LAN com bridge. Deverá ser tão fácil mover estações pela LAN com bridge quanto em uma LAN isolada.

É surpreendente como realmente é possível criar bridges transparentes. Dois algoritmos são utilizados: um algoritmo de aprendizado reverso, para evitar que o tráfego seja enviado para onde não é necessário, e um algoritmo spanning tree, para interromper loops que possam ser formados quando os switches são conectados de forma incorreta. Agora, vejamos esses algoritmos, um por vez, para aprender como essa mágica é realizada.

4.8.2 Learning Bridges

A topologia de duas LANs unidas por bridge aparece na Figura 4.38 para dois casos. No lado esquerdo, duas LANs multidrop, como as Ethernets clássicas, são unidas por uma estação especial — a bridge — que fica nas duas LANs. No lado direito, as LANs com cabos ponto a ponto, incluindo um hub, são reunidas. As bridges são os dispositivos aos quais as estações e o hub são conectados. Se a tecnologia de LAN é Ethernet, as bridges são mais bem conhecidas como switches Ethernet (conjunto de bridges em paralelo).

As bridges foram desenvolvidas quando as Ethernets clássicas estavam sendo usadas, de modo que normalmente aparecem em topologias com cabos multidrop, como na Figura 4.38(a). Porém, todas as topologias que são encontradas hoje são compostas de cabos e switches ponto a ponto. As bridges funcionam da mesma maneira nas duas configurações. Todas as estações conectadas à mesma porta em uma bridge pertencem ao mesmo domínio de colisão, e isso é diferente do domínio de colisão para outras portas. Se houver mais de uma estação, como em uma Ethernet clássica, um hub ou um enlace half-duplex, o protocolo CSMA/CD é usado para transmitir quadros.

Porém, há uma diferença no modo como são montadas as LANs conectadas com bridges. Para unir LANs multidrop, uma bridge é acrescentada como uma nova estação em cada uma das LANs multidrop, como na Figura 4.38(a). Para unir LANs ponto a ponto, os hubs são conectados a uma bridge ou, de preferência, substituídos por uma bridge, para aumentar o desempenho. Na Figura 4.38(b), as bridges substituíram todos menos um hub.

Diferentes tipos de cabos também podem ser conectados a uma bridge. Por exemplo, o cabo que conecta a bridge *B1* à *B2* na Figura 4.38(b) poderia ser um enlace de fibra óptica de longa distância, enquanto o cabo que conecta as bridges às estações poderia ser um cabo par trançado em curta distância. Esse arranjo é útil para unir LANs em prédios diferentes.

Agora, vamos considerar o que acontece dentro das bridges. Cada uma delas opera em modo promíscuo, ou seja, aceita cada quadro transmitido pelas estações conec-

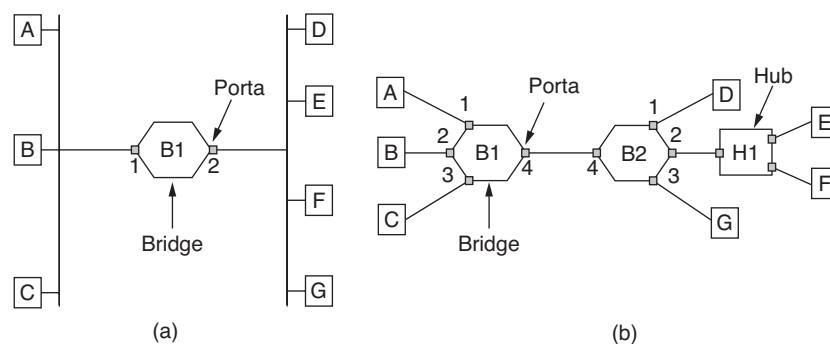



Figura 4.38 | (a) Bridge conectando duas LANs multidrop. (b) Bridges (e um hub) conectando sete estações ponto a ponto.

tadas a cada uma das portas. A bridge precisa decidir se encaminhará ou descartará cada quadro e, no primeiro caso, a que porta enviará o quadro. Essa decisão é feita usando o endereço de destino. Como exemplo, considere a topologia da Figura 4.38(a). Se a estação *A* enviar um quadro à estação *B*, a bridge *B1* receberá o quadro na porta 1. Esse quadro pode ser descartado imediatamente, sem mais cerimônias, pois já está na porta correta. Contudo, na topologia da Figura 4.38(b), suponha que *A* envie um quadro para *D*. A bridge *B1* receberá o quadro na porta 1 e o enviará pela porta 4. A bridge *B2* receberá, então, o quadro em sua porta 4 e o enviará pela sua porta 1.

Um modo simples de implementar esse esquema é ter uma grande tabela (hash) dentro da bridge. A tabela pode listar cada destino possível e a que porta de saída ele pertence. Por exemplo, na Figura 4.38(b), a tabela em *B1* listaria *D* como pertencente à porta 4, pois tudo o que *B1* precisa saber é em que porta colocar os quadros para alcançar *D*. Ou seja, na verdade, encaminhamentos posteriores ocorrerão, caso o quadro que alcança *B2* não seja de interesse para *B1*.

Quando as bridges são conectadas pela primeira vez, todas as tabelas de hash estão vazias. Nenhuma das bridges sabe onde estão os destinatários e, por isso, elas usam o algoritmo de inundação: cada quadro de entrada para um destino desconhecido é enviado para todas as LANs às quais a bridge está conectada, com exceção da LAN de que ele veio. Com o passar do tempo, as bridges aprendem onde estão os destinatários. A partir do momento em que um destinatário se torna conhecido, os quadros destinados a ele são colocados somente na porta apropriada e não são mais inundados para todas as redes.

O algoritmo usado pelas bridges é o de **aprendizado reverso**. Como já dissemos, as bridges operam em modo promíscuo; portanto, elas veem todo quadro enviado em qualquer uma de suas portas. Examinando o endereço de origem, elas podem descobrir que máquina está acessível em quais portas. Por exemplo, se a bridge *B1* da Figura 4.38(b) vir um quadro na porta 3 vindo de *C*, ela saberá que *C* pode ser alcançada através da porta 3; assim, ela cria uma entrada em sua tabela de hash. Qualquer quadro subsequente endereçado a *C* que chegue na *B1* ou em qualquer outra porta será encaminhado para a porta 3.

A topologia pode ser alterada à medida que máquinas e bridges são ativadas, desativadas e deslocadas. Para tratar de topologias dinâmicas, sempre que uma entrada de tabela de hash é criada, o tempo de chegada do quadro é indicado na entrada. Sempre que chega um quadro cuja origem já está na tabela, sua entrada é atualizada com a hora atual. Desse modo, o tempo associado a cada entrada informa a última vez que um quadro proveniente dessa máquina foi visto.

Periodicamente, um processo na bridge varre a tabela de hash e elimina todas as entradas que tenham mais de alguns minutos. Dessa forma, se um computador for desconectado de sua LAN, levado para outro lugar no prédio e reconectado nesse outro local, dentro de poucos minutos ele estará de volta à operação normal, sem nenhuma intervenção manual. Esse algoritmo também significa que, se uma máquina estiver inativa por alguns minutos, qualquer tráfego enviado a ela terá de ser difundido por inundação, até que ela mesma envie um quadro em seguida.

O procedimento de roteamento para um quadro de entrada depende da porta em que ele chega (a porta de origem) e o endereço ao qual ele se destina (o endereço de destino). O procedimento é o seguinte:

1. Se a porta para o endereço de destino e a porta de origem forem uma só, o quadro será descartado.
2. Se a porta para o endereço de destino e a porta de origem forem diferentes, o quadro será encaminhado para a porta de destino.
3. Se a porta de destino for desconhecida, o quadro será inundado e enviado para todas as portas, com exceção da porta de origem.

Você pode estar questionando se o primeiro caso pode ocorrer com enlaces ponto a ponto. A resposta é que ele pode ocorrer se forem usados hubs para conectar um grupo de computadores a uma bridge. Um exemplo aparece na Figura 4.38(b), em que as estações *E* e *F* estão conectadas ao hub *H1*, que por sua vez está conectado à bridge *B2*. Se *E* envia um quadro para *F*, o hub o repassará para *B2*, bem como para *F*. É isso que os hubs fazem — conectam todas as portas de modo que um quadro que chega a uma porta é simplesmente enviado por todas as outras portas. O quadro chegará a *B2* na porta 4, que já é a porta de saída certa para alcançar o destino. A bridge *B2* só precisa descartar o quadro.

À medida que cada quadro chegar, esse algoritmo será aplicado, de modo que ele normalmente é implementado com chip VLSI de uso especial. Os chips pesquisam e atualizam a entrada na tabela, em alguns microssegundos. Como as bridges só examinam os endereços MAC para decidir como encaminhar os quadros, é possível começar a encaminhar assim que o campo do cabeçalho de destino tiver chegado, antes que o restante do quadro tenha chegado (é claro, desde que a linha de saída esteja disponível). Esse projeto reduz a latência da passagem pela bridge, bem como o número de quadros que a bridge terá de manter em buffer. Ele é conhecido como **comutação cut-through** ou **roteamento wormhole**, e normalmente é tratado no hardware.

Podemos ver a operação de uma bridge em termos de pilhas de protocolo para entender o que significa ser um dispositivo da camada de enlace. Considere um quadro enviado da estação *A* para a estação *D* na configuração da Figura 4.38(a), em que as LANs são Ethernet. O quadro passará por uma bridge. A visão de processamento da pilha de protocolos aparece na Figura 4.39.

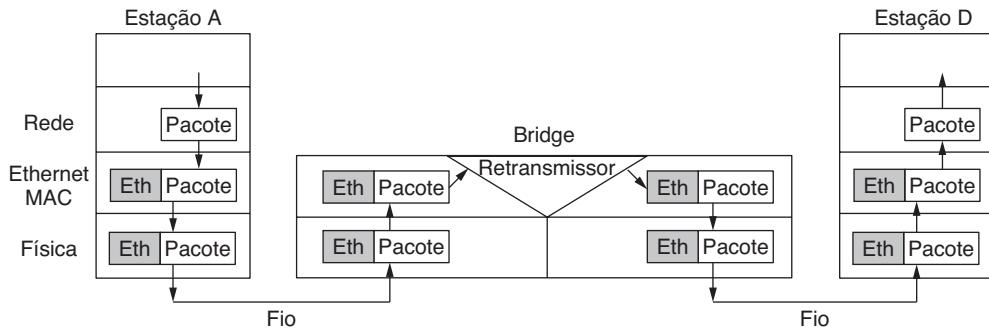



Figura 4.39 | Processamento de protocolos em uma bridge.

O pacote vem de uma camada mais alta e desce até a camada Ethernet MAC. Ele adquire um cabeçalho Ethernet (e também um final, não mostrado na figura). Essa unidade é passada para a camada física, sai pelo cabo e é apanhada pela bridge.

Na bridge, o quadro é passado da camada física para a camada Ethernet MAC. Essa camada estende o processamento em comparação com a camada Ethernet MAC em uma estação. Ela passa o quadro para um retransmissor, ainda dentro da camada MAC. O serviço de retransmissão da bridge usa apenas o cabeçalho Ethernet MAC para determinar como lidar com o quadro. Nesse caso, ela passa o quadro para a camada Ethernet MAC da porta usada para atingir a estação *D*, e o quadro continua seu caminho.

No caso geral, os retransmissores em determinada camada podem reescrever os cabeçalhos dessa camada. As VLANs oferecerão um exemplo em breve. A bridge nunca deve examinar o interior do quadro e descobrir que ele está transportando um pacote IP; isso é irrelevante para o processamento interno da bridge e violaria o uso do modelo em camadas do protocolo. Observe também que uma bridge com *k* portas terá *k* ocorrências de camadas MAC e física. O valor de *k* é 2 para nosso exemplo simples.

4.8.3 Spanning Tree Bridges

Para aumentar a confiabilidade, os enlaces redundantes podem ser usados entre as bridges. No exemplo da Figura 4.40, existem dois enlaces em paralelo entre um par de bridges. Esse projeto garante que, se um enlace for interrompido, a rede não será dividida em dois conjuntos de computadores que não podem conversar entre si.

Entretanto, essa estratégia também introduz alguns problemas adicionais, porque cria loops na topologia. Podemos ver um exemplo simples desses problemas observando como um quadro enviado por *A* para um destino previamente não observado é tratado na Figura 4.40. Cada bridge segue as regras normais para tratamento de destinos desconhecidos, que é inundar o quadro. Vamos chamar o quadro de *A* que alcança a bridge *B1* de quadro F_0. A bridge envia cópias desse quadro por todas as suas outras portas. Só consideraremos as portas da bridge que conectam *B1* a *B2* (embora o quadro também seja enviado para as outras portas). Como existem dois enlaces de *B1* para *B2*, duas cópias do quadro alcançarão *B2*. Elas são mostradas na Figura 4.40 como F_1 e F_2.

Pouco tempo depois, a bridge *B2* recebe esses quadros. Contudo, ela não sabe (e não pode saber) que eles são cópias do mesmo quadro, em vez de dois quadros diferentes enviados um após o outro. Assim, a bridge *B2* apanha F_1 e envia cópias dele para todas as outras portas, e também apanha F_2 e envia cópias dele por todas as outras portas. Isso produz quadros F_3 e F_4, enviados ao longo dos dois enlaces para *B1*. A bridge *B1*, então, vê dois novos quadros com destinos desconhecidos e os copia novamente. Esse ciclo prossegue indefinidamente.

A solução para essa dificuldade é estabelecer a comunicação entre as bridges e sobrepor a topologia real com uma spanning tree que alcance cada bridge. Na realidade, algumas conexões potenciais entre as bridges são ignoradas para que se construa uma topologia virtual livre de loops, que é um subconjunto da topologia real.

Por exemplo, na Figura 4.41, vemos cinco bridges interconectadas que também têm estações conectadas a elas. Essa estação se conecta a apenas uma bridge. Existem algumas conexões redundantes entre as bridges para que os quadros sejam encaminhados em loops se todos os enlaces forem usados. Essa topologia pode ser considerada como um grafo em que as bridges são os nós e os enlaces ponto a ponto são as arestas. O grafo pode ser reduzido

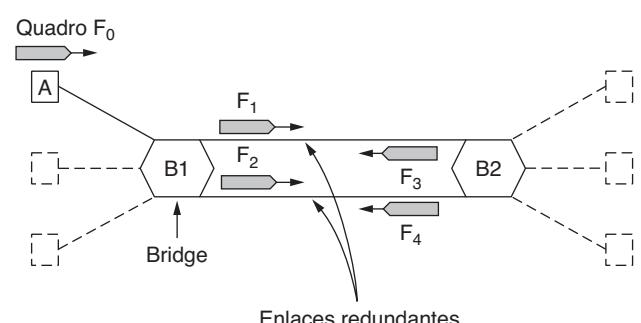



Figura 4.40 | Bridges com dois enlaces paralelos.

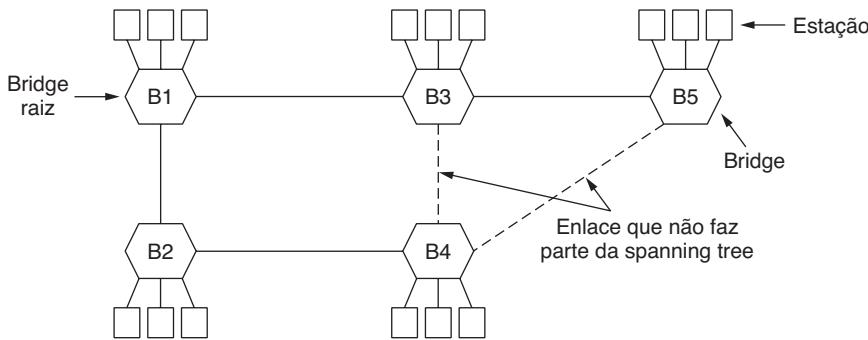



Figura 4.41 | Uma spanning tree conectando cinco bridges. As linhas pontilhadas não fazem parte da spanning tree.

a uma spanning tree, que não tem ciclos, por definição, eliminando os arcos mostrados como linhas tracejadas na Figura 4.41. Com a utilização da spanning tree, existe exatamente um caminho de cada estação para qualquer outra estação. Uma vez que as bridges entram em acordo em relação à spanning tree, tudo o que é encaminhado entre as estações segue a spanning tree. Como existe um único caminho de cada origem até cada destino, é impossível haver loops.

Para construir a spanning tree, as bridges executam um algoritmo distribuído. Cada bridge transmite periodicamente por broadcast uma mensagem de configuração por todas as suas portas aos seus vizinhos e processa as mensagens que recebe de outras bridges, conforme descreveremos a seguir. Essas mensagens não são encaminhadas, pois seu propósito é construir a árvore, que pode, então, ser usada para o encaminhamento.

Primeiro as bridges precisam escolher entre elas, a que será usada como raiz da spanning tree. Para fazer essa escolha, cada uma delas inclui um identificador com base no endereço MAC na mensagem de configuração, bem como o identificador da bridge que se acredita ser a raiz. Os endereços MAC são instalados pelo fabricante com a garantia de ser exclusivos em todo o mundo, o que torna esses identificadores convenientes e exclusivos. As bridges escolhem aquela com o menor identificador para ser a raiz. Depois que mensagens suficientes tiverem sido trocadas para espalhar a notícia, todas as bridges chegarão a um acordo sobre qual bridge é a raiz. Na Figura 4.41, a bridge *B1* tem o menor identificador e se torna a raiz.

Em seguida, é construída uma árvore de caminhos mais curtos a partir da raiz até cada bridge. Na Figura 4.41, as bridges *B2* e *B3* podem ser alcançadas diretamente a partir da *B1*, em um salto que é o caminho mais curto. A *B4* pode ser alcançada em dois saltos, por meio de *B2* ou *B3*. Para desempatar, é escolhido o caminho por meio da bridge com o menor identificador, de modo que *B4* é alcançada por meio de *B2*. A *B5* pode ser alcançada em dois saltos por meio de *B3*.

Para encontrar esses caminhos mais curtos, as bridges incluem a distância a partir da raiz em suas mensagens de configuração. Cada uma delas se lembra do caminho mais curto que encontra até a raiz. As bridges, então, desativam as portas que não fazem parte do caminho mais curto.

Embora a árvore se espalhe por todas as bridges, nem todos os enlaces (ou mesmo as bridges) estão necessariamente presentes na árvore. Isso acontece porque o desligamento das portas poda alguns enlaces da rede e impede loops. Mesmo depois que a spanning tree é estabelecida, o algoritmo continua a funcionar durante a operação normal, com a finalidade de detectar automaticamente mudanças na topologia e atualizar a árvore.

O algoritmo distribuído usado para a construção da spanning tree foi inventado por Radia Perlman. Seu trabalho foi resolver o problema de juntar LANs sem loops. Ela teve uma semana para fazer isso, mas teve a ideia do algoritmo spanning tree em um dia. Felizmente, ela teve tempo suficiente para escrevê-lo em forma de poema (Perlman, 1985):

I think that I shall never see
A graph more lovely than a tree.
A tree whose crucial property
Is loop-free connectivity.
A tree which must be sure to span.
So packets can reach every LAN.
First the Root must be selected
By ID it is elected.
Least cost paths from Root are traced
In the tree these paths are placed.
A mesh is made by folks like me
Then bridges find a spanning tree.

(Creio que nunca verei/Um grafo melhor que uma árvore./Uma árvore cuja propriedade fundamental/É a conectividade sem loops./Uma árvore que precisa se espalhar/Para que os pacotes possam alcançar cada LAN./Primeiro a raiz deve ser selecionada/Por ID ela é eleita./Caminhos de menor custo a partir da raiz são traçados./Na árvore esses caminhos são colocados./Uma malha é feita por pessoas como eu/Depois as bridges acham uma spanning tree.)

O algoritmo spanning tree foi, então, padronizado como IEEE 802.1D e usado por muitos anos. Em 2001, ele

foi revisado para encontrar mais rapidamente uma nova spanning tree após uma mudança de topologia. Para ver um estudo mais detalhado sobre as bridges, consulte Perlman (2000).

4.8.4 Repetidores, hubs, bridges, switches, roteadores e gateways

Até agora neste livro, examinamos diversas maneiras de transferir quadros e pacotes de um computador para outro. Mencionamos repetidores, hubs, bridges, switches, roteadores e gateways. Todos esses dispositivos são de uso comum, mas diferem entre si em detalhes sutis e não muito sutis. Por existir uma grande quantidade desses dispositivos, provavelmente vale a pena examiná-los em conjunto para ver quais são as semelhanças e as diferenças entre eles.

A chave para entender esses dispositivos é observar que eles operam em camadas diferentes, como ilustra a Figura 4.42(a). A camada é importante, porque diferentes dispositivos utilizam fragmentos de informações diferentes para decidir como realizar a comutação. Em um cenário típico, o usuário gera alguns dados a ser enviados para uma máquina remota. Esses dados são repassados à camada de transporte, que então acrescenta um cabeçalho (por exemplo, um cabeçalho TCP) e repassa o pacote resultante à camada de rede situada abaixo dela. Essa camada adiciona seu próprio cabeçalho para formar um pacote da camada de rede (por exemplo, um pacote IP). Na Figura 4.42(b), vemos o pacote IP sombreado na cor cinza. Em seguida, o pacote vai para a camada de enlace de dados, que adiciona seu próprio cabeçalho e seu checksum (CRC) e entrega o quadro resultante à camada física para transmissão, digamos, por uma LAN.

Agora, vamos examinar os dispositivos de comutação e ver como eles se relacionam aos pacotes e quadros. Na parte inferior, na camada física, encontramos os repetidores. Estes são dispositivos analógicos que trabalham com sinais nos cabos aos quais estão conectados. Um sinal que aparece em um deles é limpo, amplificado e colocado em outro cabo. Os repetidores não reconhecem quadros, pacotes ou cabeçalhos. Eles entendem os símbolos codificados em bits como volts. Por exemplo, a Ethernet clássica foi projetada para permitir quatro repetidores, que amplificam o sinal a fim de estender o comprimento máximo do cabo de 500 para 2.500 metros.

Em seguida, temos os hubs. Um hub tem várias interfaces de entrada que ele conecta eletricamente. Os quadros que chegam a quaisquer dessas interfaces são enviados a todas as outras. Se dois quadros chegarem ao mesmo tempo, eles colidirão, exatamente como ocorre em um cabo coaxial. Todas as linhas que chegam a um hub devem operar na mesma velocidade. Os hubs diferem dos repetidores pelo fato de (normalmente) não amplificarem os sinais de entrada e ser projetados para conter várias linhas de entrada, mas as diferenças são pequenas. Assim como os repetidores, os hubs são dispositivos da camada física que não examinam os endereços da camada de enlace nem os utilizam de maneira alguma.

Agora, vamos subir até a camada de enlace de dados, em que encontramos bridges e switches. Acabamos de estudar as bridges com certa profundidade. Uma bridge conecta duas ou mais LANs. Como um hub, uma bridge moderna tem várias portas, normalmente o suficiente para 4 a 48 linhas de entrada de um certo tipo. Diferentemente de um hub, cada porta é isolada para ser seu próprio domínio de colisão; se a porta tem uma linha ponto a ponto full-duplex, o algoritmo CSMA/CD não é necessário. Quando um quadro chega, a ponte extrai o endereço de destino do cabeçalho de quadro e examina uma tabela, a fim de verificar para onde deve enviá-lo. No caso de uma rede Ethernet, esse endereço é o endereço de destino de 48 bits mostrado na Figura 4.14. A bridge só envia o quadro à porta onde ele é necessário, e pode encaminhar vários quadros ao mesmo tempo.

As bridges oferecem desempenho muito melhor que os hubs, e o isolamento entre suas portas também significa que as linhas de entrada podem trabalhar com diferentes velocidades, possivelmente ainda com diferentes tipos de rede. Um exemplo comum é uma bridge com portas que se conectam à Ethernet de 10, 100 e 1.000 Mbps. O uso de buffer dentro da bridge é necessário para aceitar um quadro em uma porta e transmitir o quadro por uma porta

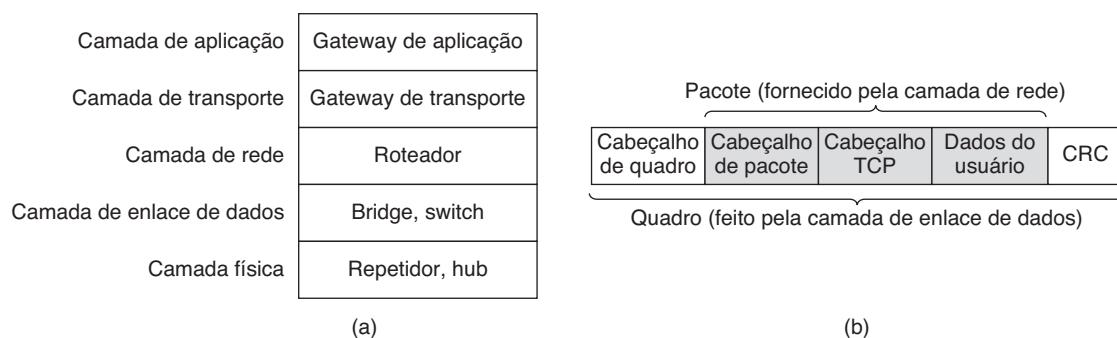



Figura 4.42 | (a) Dispositivos presentes em cada camada. (b) Quadros, pacotes e cabeçalhos.

diferente. Se os quadros entrarem mais rapidamente do que podem ser retransmitidos, a bridge poderá ficar sem espaço em buffer e ter de começar a descartar quadros. Por exemplo, se uma gigabit Ethernet estiver empurrando bits para uma Ethernet de 10 Mbps na velocidade máxima, a bridge terá de mantê-los em buffer, na esperança de não ficar sem memória. Esse problema ainda existe mesmo que todas as portas trabalhem na mesma velocidade, pois mais de uma porta pode estar enviando quadros a determinada porta de destino.

As bridges visavam originalmente à união de diferentes tipos de LANs, por exemplo, uma LAN Ethernet e uma Token Ring. Contudo, isso nunca funcionou bem, em razão das diferenças entre as LANs. Diferentes formatos de quadro exigem cópia e reformatação, o que requer tempo de CPU, um novo cálculo de checksum e introduz a possibilidade de erros não detectados, em decorrência de bits ruins na memória da bridge. O uso de diferentes tamanhos máximos de quadro também é um problema sério sem uma boa solução. Basicamente, quadros muito grandes para ser encaminhados devem ser descartados. Muita coisa para se evidenciar.

As duas áreas em que as LANs podem diferir são segurança e qualidade de serviço. Algumas LANs têm criptografia da camada de enlace (por exemplo, 802.11) e outras não (por exemplo, Ethernet). Algumas LANs têm recursos de qualidade de serviço, como prioridades (por exemplo, 802.11) e outras não (por exemplo, Ethernet). Consequentemente, quando um quadro precisa trafegar entre essas LANs, pode não ser possível fornecer a segurança ou a qualidade de serviço esperadas pelo emissor. Por todos esses motivos, as bridges modernas normalmente funcionam para um tipo de rede, e os roteadores, que veremos em breve, são usados em seu lugar para unir redes de diferentes tipos.

Os switches são bridges modernas com outro nome (na verdade, um conjunto de bridges forma um switch). As diferenças são mais por questões de marketing do que técnicas, mas existem alguns pontos que precisam ser conhecidos. As bridges foram desenvolvidas quando a Ethernet clássica estava em uso, de modo que tendem a unir relativamente poucas LANs e, portanto, ter relativamente poucas portas. O termo 'switch' é mais popular hoje em dia. Além disso, todas as instalações modernas usam enlaces ponto a ponto, como cabos de par trançado, de modo que computadores individuais se conectam diretamente a um switch e, portanto, este costuma ter muitas portas. Finalmente, 'switch' também é usado como um termo geral. Com uma bridge, a funcionalidade é clara. Por outro lado, um switch pode se referir a um switch Ethernet ou a um tipo de dispositivo completamente diferente que toma decisões de encaminhamento, como um switch usado em telefonia.

Até o momento, vimos repetidores e hubs, que são bastante semelhantes, bem como bridges e switches, que também são bem parecidos. Agora vamos passar para os roteadores, os quais são diferentes de todos os dispositivos anteriores. Quando um pacote entra em um roteador, o cabeçalho de quadro e o final são retirados, e o pacote localizado no campo de carga útil do quadro (sombreado na Figura 4.42) é repassado ao software de roteamento. Esse software utiliza o cabeçalho de pacote para escolher uma interface de saída. No caso de um pacote IP, o cabeçalho do pacote conterá um endereço de 32 bits (IPv4) ou de 128 bits (IPv6), mas não um endereço 802 de 48 bits. O software de roteamento não vê os endereços de quadro e nem mesmo sabe se o pacote veio de uma LAN ou de uma conexão ponto a ponto. Estudaremos os roteadores e o roteamento no Capítulo 5.

Subindo até outra camada, encontramos gateways de transporte. Esses dispositivos conectam dois computadores que utilizam diferentes protocolos de transporte orientados a conexões. Por exemplo, suponha que um computador que utiliza o protocolo TCP/IP orientado a conexões precise se comunicar com um computador que utiliza um protocolo de transporte orientado a conexões diferente, chamado SCTP. O gateway de transporte pode copiar os pacotes de uma conexão para a outra, reformatando-os caso seja necessário.

Finalmente, os gateways de aplicação reconhecem o formato e o conteúdo dos dados e convertem mensagens de um formato para outro. Por exemplo, um gateway de correio eletrônico poderia converter mensagens da Internet em mensagens SMS para telefones móveis. Assim como 'switch', 'gateway' é um termo bastante genérico. Ele se refere a um processo de encaminhamento que atua em uma camada superior.

4.8.5 LANs virtuais

Quando foram criadas as primeiras redes locais, grossos cabos amarelos se estendiam pelos conduítes de muitos prédios comerciais. Todo computador era conectado a esses cabos por onde eles passavam. Ninguém parava para pensar sobre que computador pertencia a qual LAN. Todas as pessoas que trabalhavam em escritórios adjacentes tinham seus equipamentos conectados à mesma LAN, quer elas trabalhassem juntas, quer não. A geografia era mais importante que os organogramas corporativos.

Com o advento do par trançado e dos hubs na década de 90, tudo isso mudou. A fiação dos prédios foi trocada (a um custo considerável) para eliminar todos os grossos cabos amarelos e instalar pares trançados, que iam de cada escritório até os armários centrais de fiação instalados no fim de cada corredor ou em uma sala de máquinas central, como ilustra a Figura 4.43. Se o encarregado da fiação fosse um visionário, eram instalados pares trançados da Categoria 5; se fosse um avarento, a fiação telefônica existente (da Categoria 3) era usada (até ser substituída alguns anos mais tarde, quando surgiu a Fast Ethernet).

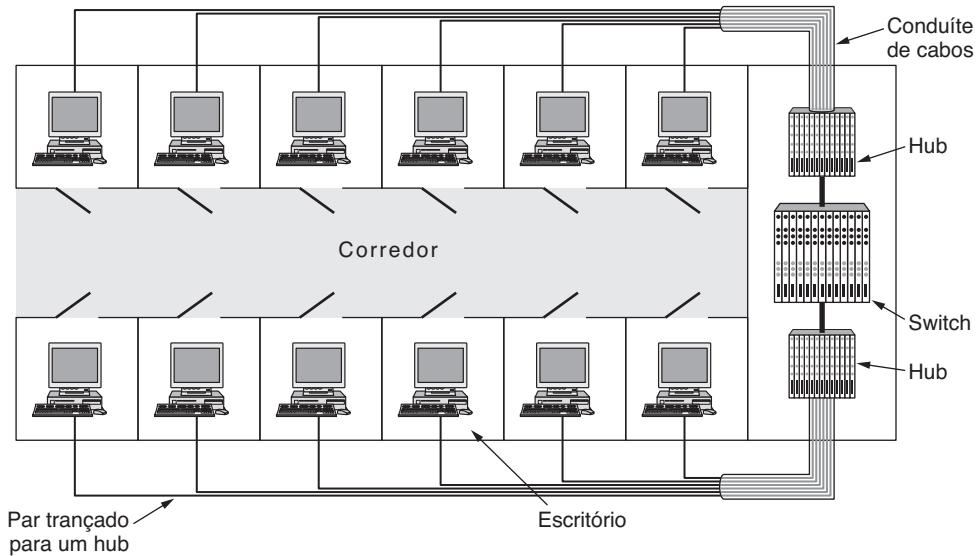



Figura 4.43 | Um prédio com fiação centralizada, utilizando hubs e um switch.

Hoje, os cabos mudaram e os hubs se tornaram switches, mas o padrão de fiação ainda é o mesmo. Esse padrão possibilita configurar LANs logicamente, em vez de fisicamente. Por exemplo, se uma empresa deseja *k* LANs, ela pode comprar *k* switches. Escolhendo cuidadosamente quais conectores ligar a quais switches, os ocupantes de uma LAN podem ser escolhidos de um modo que faça sentido para a organização, sem considerar muito a geografia.

É importante saber quem está conectado a cada LAN? Afinal, em quase todas as organizações, todas as LANs estão interconectadas. A resposta é sim, isso com frequência é importante. Os administradores de redes gostam de agrupar os usuários em LANs de modo a refletir a estrutura organizacional, em lugar do layout físico do prédio, por várias razões. Uma delas é a segurança. Uma LAN poderia hospedar servidores Web e outros computadores voltados para uso público. Outra LAN poderia hospedar computadores que contivessem os registros do departamento de recursos humanos, que não devem ser passados para fora do departamento. Nessa situação, faz sentido colocar todos os computadores em uma única LAN e não permitir que nenhum servidor seja acessado de fora da LAN. A gerência tende a franzir a testa quando escuta que esse arranjo é impossível.

Uma segunda questão é a carga. Algumas LANs são utilizadas mais intensamente que outras, e pode ser interessante separá-las. Por exemplo, se o pessoal da área de pesquisa estiver realizando várias experiências e alguma delas sair do controle e saturar a LAN, é bem possível que o pessoal da gerência não fique muito entusiasmado por ter de doar uma parte de sua capacidade de computação que estava usando para uma videoconferência para ajudar os colegas do outro departamento. Novamente, isso pode causar, na gerência, a impressão da necessidade de instalar uma rede mais rápida.

Uma terceira questão é o tráfego de broadcast. As bridges enviam tráfego de broadcast quando o local do destino é desconhecido, e os protocolos da camada superior também usam o broadcast. Por exemplo, quando um usuário quer enviar um pacote a um endereço IP representado por *x*, como saber qual endereço MAC colocar no quadro? Estudaremos essa questão no Capítulo 5, mas, em resumo, o usuário transmitirá um quadro contendo a seguinte pergunta: "A quem pertence o endereço IP *x*?". Em seguida, o usuário aguardará uma resposta. À medida que o número de comunicações em uma LAN aumenta, o mesmo acontece com a quantidade de broadcasts circulando. Cada broadcast consome mais capacidade da LAN do que um quadro normal, pois ele é entregue a cada computador na LAN. Evitando que as LANs sejam maiores do que precisam ser, reduzimos o impacto do tráfego de broadcast.

Um problema relacionado ao broadcast é que, de vez em quando, uma interface de rede sofrerá uma pane e começará a gerar um fluxo infinito de quadros de broadcast. Se a rede realmente estiver sem sorte, alguns desses quadros gerarão respostas que levarão a ainda mais tráfego. O resultado dessa **tempestade de broadcast** é que (1) a capacidade da LAN inteira será ocupada por esses quadros e (2) todas as máquinas em todas as LANs interconectadas serão danificadas, processando e descartando todos os quadros que estiverem sendo transmitidos.

A princípio, pode parecer que seria possível limitar o escopo das tempestades de broadcast separando as LANs com bridges ou switches; porém, se o objetivo é conseguir transparência (isto é, poder mover uma máquina de uma LAN diferente usando a bridge sem que alguém note a mudança), então as bridges têm de encaminhar quadros de broadcast.

Depois de verificarmos por que seria interessante para as empresas ter várias LANs com escopo restrito, vamos voltar ao problema de desacoplar a topologia lógica da topologia física. A criação de uma topologia física para refletir a estrutura organizacional pode acrescentar trabalho e custo, mesmo com fiação centralizada e switches. Por exemplo, se duas pessoas no mesmo departamento trabalham em prédios diferentes, pode ser mais fácil conectá-las a diferentes switches que fazem parte de LANs diferentes. Mesmo que esse não seja o caso, um usuário poderia ser deslocado dentro da empresa de um departamento para outro sem mudar de escritório, ou poderia mudar de escritório sem mudar de departamento. Isso pode fazer com que o usuário esteja na LAN errada até que o administrador mude o conector do usuário de um switch para outro. Além disso, o número de computadores que pertencem a diferentes departamentos pode não corresponder bem ao número de portas nos switches; alguns departamentos podem ser muito pequenos e outros tão grandes que exigem vários switches. Isso resulta em portas do switch desperdiçadas, que não são usadas.

Em muitas empresas, as mudanças organizacionais ocorrem o tempo todo; isso significa que os administradores de sistemas passam muito tempo retirando plugues e inserindo-os de novo em algum outro lugar. Além disso, em alguns casos, a mudança não pode ser feita de modo algum, porque o par trançado da máquina do usuário está longe demais do hub correto (por exemplo, em outro prédio), ou então as portas do switch disponíveis estão na LAN errada.

Em resposta à solicitação de usuários que desejam maior flexibilidade, os fornecedores de redes começaram a buscar um meio de recompor a fiação dos prédios inteiramente via software. O conceito resultante é chamado LAN virtual, ou **VLAN** (**Virtual LAN**) e foi até mesmo padronizado pelo comitê IEEE 802. Atualmente, ele está sendo implementado em muitas organizações. Para ver mais informações sobre as VLANs, consulte Seifert e Edwards (2008).

As VLANs se baseiam em switches especialmente projetados para reconhecê-las. Para configurar uma rede baseada em VLANs, o administrador da rede decide quantas delas haverá, quais computadores estarão em qual VLAN e qual será o nome de cada uma. Geralmente, as VLANs são identificadas (informalmente) por cores, pois assim é possível imprimir diagramas de cores que mostram o layout físico das máquinas, com os membros da LAN vermelha em vermelho, os membros da LAN verde em verde, e assim por diante. Desse modo, os layouts físico e lógico são visíveis em um único diagrama.

Como exemplo, considere as LANs conectadas por bridges da Figura 4.44, em que nove das máquinas pertencem à VLAN G (*gray* — cinza) e cinco pertencem à VLAN W (*white* — branca). As máquinas da VLAN cinza estão espalhadas por dois switches, incluindo as duas máquinas que se conectam a um switch por meio de um hub.

Para fazer as VLANs funcionar corretamente, é necessário definir tabelas de configuração nas bridges ou nos switches. Essas tabelas informam quais são as VLANs acessíveis através de cada uma das portas. Quando um quadro chega, digamos, da VLAN cinza, ele tem de ser encaminhado para todas as portas marcadas com um G. Isso é válido para o tráfego comum (isto é, de unicast) para o qual as pontes não descobriram o local do destino, bem como para os tráfegos de multicast e de broadcast. Observe que uma porta pode ser rotulada com várias cores de VLAN.

Como um exemplo, suponha que uma das estações cinza conectadas à bridge *B1* na Figura 4.44 envie um quadro para um destino que não tenha sido observado anteriormente. A bridge *B1* receberá o quadro e verá que ele veio de uma máquina na VLAN cinza e, por isso, inundará o quadro em todas as portas rotuladas com G (exceto a porta de chegada). O quadro será enviado às cinco outras estações cinza conectadas a *B1*, além do enlace de *B1* até a bridge *B2*. Na *B2*, o quadro será igualmente encaminhado por todas as portas rotuladas com G. Isso envia o quadro a uma estação adiante e ao hub (que transmitirá o quadro a todas as suas estações). O hub tem os dois rótulos, pois se conecta a máquinas das duas VLANs. O quadro não é enviado pelas outras portas sem G no rótulo, pois a bridge sabe que não existem máquinas na VLAN cinza que possam ser alcançadas por meio dessas portas.

Em nosso exemplo, o quadro é enviado apenas da bridge *B1* para a bridge *B2*, pois existem máquinas na VLAN cinza que estão conectadas a *B2*. Examinando a VLAN branca, podemos ver que a porta da *B2* que se conecta à *B1*

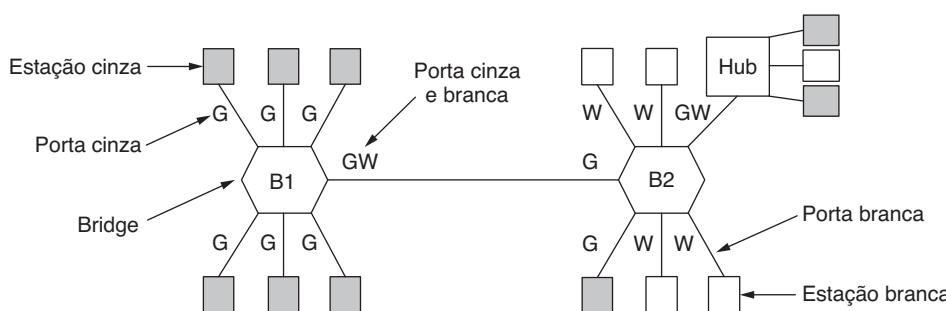



Figura 4.44 | Duas VLANs, cinza e branca, em uma LAN com bridge.

não está rotulada com W. Isso significa que um quadro na VLAN branca não será encaminhado da *B2* para a *B1*. Esse comportamento é correto, pois nenhuma estação na VLAN branca está conectada à *B1*.

O padrão IEEE 802.1Q

Para implementar esse esquema, as bridges precisam saber a qual VLAN pertence um quadro que chega. Sem essa informação, por exemplo, quando a bridge *B2* receber um quadro da bridge *B1*, na Figura 4.44, ela não poderá saber se encaminhará o quadro na VLAN cinza ou branca. Se estivéssemos criando um novo tipo de LAN, seria muito fácil apenas acrescentar um campo VLAN no cabeçalho. Mas o que fazer no caso do padrão Ethernet, que é a LAN dominante e que não tem um campo sobressalente que possa ser usado como identificador da VLAN?

O comitê 802 do IEEE enfrentou esse problema em 1995. Depois de muita discussão, ele fez o inconcebível e mudou o cabeçalho do padrão Ethernet. O novo formato foi publicado no padrão **802.1Q** do IEEE, emitido em 1998. O novo formato contém uma tag de VLAN; vamos examiná-lo rapidamente. Não surpreende que a mudança de algo tão bem estabelecido quanto o cabeçalho Ethernet não seja inteiramente trivial. Algumas questões que surgem são:

1. Precisaremos jogar fora várias centenas de milhões de placas Ethernet existentes?
2. Se não, quem gerará os novos campos?
3. O que acontecerá com os quadros que já têm o tamanho máximo?

É claro que o comitê 802 estava (ainda que de forma muito dolorosa) consciente desses problemas e teve de apresentar soluções, o que realmente fez.

A chave para a solução é perceber que os campos VLAN só são realmente usados pelas bridges e switches, e *não* pelas máquinas dos usuários. Desse modo, na Figura 4.44, não é realmente essencial que eles estejam presentes nas linhas que saem para as estações finais, desde que estejam na linha entre as bridges ou os switches. Portanto, para usar VLANs, as bridges têm de reconhecer a VLAN. Esse fato torna o projeto viável.

Quanto a descartar todas as placas Ethernet existentes, a resposta é não. Lembre-se de que o comitê 802.3 não poderia nem mesmo fazer as pessoas transformar o campo *Tipo* em um campo *Tamanho*. Você pode imaginar a reação ao anúncio de que todas as placas Ethernet existentes teriam de ser jogadas fora. Porém, à medida que novas placas Ethernet entrarem no mercado, espera-se que elas sejam compatíveis com o 802.1Q e possam preencher corretamente os campos VLAN.

Como pode haver computadores (e switches) que não reconhecem a VLAN, a primeira bridge que a reconhece e toca em um quadro acrescenta os campos de VLAN e a última no caminho os remove. Um exemplo de topologia mista aparece na Figura 4.45. Nela, os computadores que reconhecem a VLAN geram quadros marcados (ou seja, 802.1Q) diretamente, e a comutação posterior utiliza essas tags. Os símbolos sombreados são máquinas que reconhecem VLANs e os símbolos vazios não as reconhecem.

Com o 802.1Q, os quadros são coloridos dependendo da porta na qual são recebidos. Para que esse método funcione, todas as máquinas em uma porta precisam pertencer à mesma VLAN, o que reduz a flexibilidade. Por exemplo, na Figura 4.45, essa propriedade é mantida para todas as portas nas quais um computador individual se conecta a uma bridge, mas não para a porta na qual o hub se conecta a *B2*.

Além disso, a bridge pode usar o protocolo da camada mais alta para selecionar a cor. Desse modo, os quadros que chegam a uma porta podem ser colocados em VLANs diferentes, dependendo se elas transportam pacotes IP ou quadros PPP.

Outros métodos são viáveis, mas não são admitidos pelo 802.1Q. Como exemplo, o endereço MAC pode ser usado para selecionar a cor da VLAN. Isso poderia ser útil para quadros chegando de uma LAN 802.11 próxima, em que notebooks enviam quadros por portas diferentes enquanto se movem. Um endereço MAC seria, então, mapeado a uma VLAN fixa, independentemente da porta em que ele entrou na LAN.

Quanto ao problema de quadros maiores que 1.518 bytes, o 802.1Q simplesmente aumentou o limite para 1.522 bytes. Felizmente, apenas computadores e switches

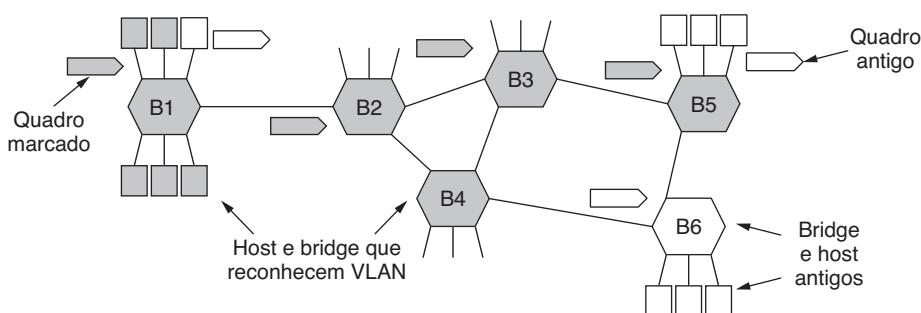



Figura 4.45 | LAN com bridge, parcialmente consciente da VLAN. Os sombreados são máquinas que reconhecem VLANs e os vazios não as reconhecem.

que reconhecem a VLAN precisam dar suporte a esses quadros maiores.

Agora, vamos examinar o formato de quadro 802.1Q. Ele está representado na Figura 4.46. A única mudança é o acréscimo de um par de campos de 2 bytes. O primeiro é o campo *ID de protocolo de VLAN*, que sempre tem o valor 0x8100. Como esse número é maior que 1.500, todas as placas Ethernet o interpretam como um tipo, e não como um tamanho. O que uma placa antiga faz com um quadro desse tipo é discutível, pois tais quadros não deveriam ser enviados a placas antigas.

O segundo campo de 2 bytes contém três subcampos. O principal é o *Identificador de VLAN*, que ocupa os 12 bits de baixa ordem. É isso que interessa — a cor da VLAN à qual o quadro pertence. O campo de 3 bits *Prioridade* não tem nenhuma relação com VLANs, mas, como a mudança no cabeçalho Ethernet é um evento que acontece uma vez a cada década, demora três anos e envolve uma centena de pessoas, por que não incluir alguns outros benefícios? Esse campo torna possível distinguir o tráfego em tempo real permanente do tráfego em tempo real provisório e do tráfego não relacionado ao tempo, a fim de fornecer melhor qualidade de serviço nas redes Ethernet. Ele é necessário para voz sobre a Ethernet (embora o IP tivesse um campo semelhante durante um quarto de século sem que ninguém jamais o tenha usado).

O último campo, o indicador de formato canônico, ou *CFI* (*Canonical Format Indicator*), deveria ter sido chamado indicador de ego corporativo, ou *CEI* (*Corporate Ego Indicator*). Originalmente, ele foi criado para indicar endereços MAC little-endian *versus* endereços MAC big-endian, mas esse uso se perdeu em outras controvérsias. Sua presença agora indica que a carga útil contém um quadro 802.5 congelado que está esperando encontrar outra LAN 802.5 no destino, enquanto é transportado por uma rede Ethernet nesse meio-tempo. É claro que toda essa organização não tem nenhuma relação com as VLANs. No entanto, a política do comitê de padrões não é diferente da política comum: se você votar a favor do meu bit, eu votarei a favor do seu.

Como mencionado anteriormente, quando um quadro marcado chega a um switch que reconhece VLANs, o switch utiliza a ID da VLAN como um índice em uma tabela, para descobrir por meio de que portas deve enviar o quadro. Porém, de onde vem a tabela? Se ela for construída manualmente, voltaremos à estaca zero: a configuração manual de bridges. A beleza da ponte transparente é o fato de ela ser plug-and-play e não exigir nenhuma configuração manual. Seria uma vergonha terrível perder essa propriedade. Felizmente, as bridges que reconhecem VLANs também podem se autoconfigurar com base na observação das tags que passam por elas. Se um quadro marcado como VLAN 4 chegar à porta 3, então aparentemente alguma máquina na porta 3 está na VLAN 4. O padrão 802.1Q explica como construir as tabelas dinamicamente, em grande parte por meio de referências a partes apropriadas do padrão 802.1D.

Antes de encerrarmos o assunto de roteamento de VLANs, vale a pena fazer uma última observação. Muitas pessoas no universo da Internet e das redes Ethernet defendem, de forma enfática, a interligação de redes não orientadas a conexões e se opõem violentamente a qualquer sinal de conexões na camada de enlace de dados ou na camada de rede. Ainda assim, as VLANs introduzem algo surpreendentemente semelhante a uma conexão. Para usar as VLANs de forma apropriada, cada quadro transporta um novo identificador especial que é usado como um índice para uma tabela interna do switch, a fim de procurar o local para onde o quadro deve ser enviado. É isso mesmo que acontece nas redes orientadas a conexões. Nas redes não orientadas a conexões, deve-se utilizar o endereço de destino para roteamento e não alguma espécie de identificador de conexão. Veremos mais detalhes sobre essa rejeição às conexões no Capítulo 5.

4.9 Resumo

Algumas redes administram todo o fluxo de comunicações por meio de um único canal. Nessas redes, a grande questão é a alocação desse canal entre as estações que desejam utilizá-lo. FDM e TDM são esquemas de alocação sim-

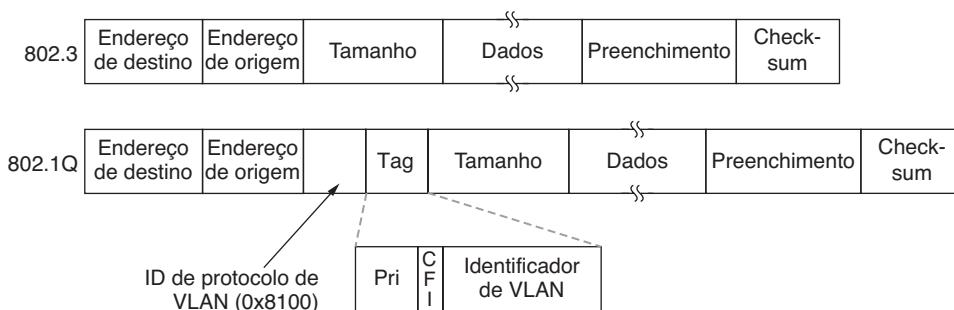



Figura 4.46 | Os formatos de quadros Ethernet 802.3 (antigo) e 802.1Q.

ples e eficientes quando o número de estações é pequeno e fixo, e o tráfego é contínuo. Ambos são amplamente utilizados nessas circunstâncias, como para dividir a largura de banda nos enlaces usados como troncos telefônicos. No entanto, quando o número de estações é grande e variável, ou quando o tráfego ocorre em rajadas — o caso comum nas redes de computadores — FDM e TDM não são boas opções.

Foram criados diversos algoritmos de alocação de canal dinâmico. O protocolo ALOHA, com e sem segmentação (o ALOHA original ou o slotted ALOHA), é usado em muitas derivações nos sistemas reais, por exemplo, modems a cabo e RFID. Como uma melhoria quando o estado do canal pode ser detectado, as estações podem evitar iniciar uma transmissão enquanto outra estação está transmitindo. Essa técnica, a detecção de portadora, levou a uma série de protocolos CSMA para LANs e MANs. Essa é a base para a Ethernet clássica e as redes 802.11.

Uma classe de protocolos que elimina por completo a disputa, ou pelo menos a reduz consideravelmente, é bastante conhecida. O protocolo bit-map, topologias em anéis e a contagem regressiva binária eliminam totalmente a disputa. O protocolo tree walk reduz a disputa dividindo de forma dinâmica as estações em dois grupos com tamanhos diferentes, e permitindo a disputa apenas dentro de um grupo; o ideal é que esse grupo seja escolhido de tal forma que apenas uma estação que esteja pronta para transmitir tenha permissão para fazê-lo.

As LANs sem fios têm os problemas adicionais de dificuldade de detectar as transmissões que colidem, e as regiões de cobertura das estações podem ser diferentes. Na LAN sem fios dominante, IEEE 802.11, as estações usam CSMA/CA para aliviar o problema de deixar pequenas lacunas para impedir colisões. As estações também podem usar o protocolo RTS/CTS para combater terminais ocultos que surgem em decorrência do segundo problema. O IEEE 802.11 normalmente é usado para conectar notebooks e outros dispositivos a pontos de acesso wireless, mas também pode ser usado entre dispositivos. Qualquer uma das várias camadas físicas pode ser usada, incluindo o FDM multicanal com e sem várias antenas, e o espectro de dispersão.

Assim como o 802.11, leitoras de RFID utilizam um protocolo de acesso aleatório para os identificadores se comunicarem. Outras PANs e MANs sem fios possuem projetos diferentes. O sistema Bluetooth conecta fones de ouvido e muitos tipos de periféricos a computadores sem a utilização de fios. O IEEE 802.16 oferece um serviço de dados de Internet sem fios remoto para computadores fixos e móveis. Essas duas redes utilizam um projeto centralizado, orientado a conexões, em que o mestre Bluetooth e a estação-base WiMAX decidem quando cada estação pode enviar ou receber dados. Para o 802.16, esse projeto admite diferentes qualidades de serviço para o tráfego em tempo real, como chamadas telefônicas e tráfego interativo, como a navegação na Web. Para o Bluetooth, colocar a complexidade no mestre possibilita o uso de dispositivos escravos mais econômicos.

A Ethernet é a forma dominante de LAN com fios. A Ethernet clássica usava CSMA/CD para alocação de canal em um cabo amarelo do tamanho de uma mangueira de jardim, esticado de uma máquina para outra. A arquitetura mudou quando as velocidades passaram de 10 Mbps para 10 Gbps, e continuam subindo. Agora, enlaces ponto a ponto, como o par trançado, são conectados a hubs e switches. Com os switches modernos e enlaces full-duplex, não existe disputa nos enlaces e o switch pode encaminhar os quadros entre diferentes portas em paralelo.

Com prédios repletos de LANs, é preciso que haja uma maneira de interconectar todos eles. As bridges plug-and-play são usadas para essa finalidade, sendo construídas com um algoritmo de aprendizado e um algoritmo spanning tree. Como essa funcionalidade está embutida nos switches modernos, os termos ‘bridge’ e ‘switch’ são usados para indicar a mesma coisa. Para ajudar no gerenciamento de LANs com bridges, as VLANs permitem que a topologia física seja dividida em diferentes topologias lógicas. O padrão de VLAN, o IEEE 802.1Q, introduz um novo formato de quadros Ethernet.

Problemas

1. Para resolver este problema, use uma fórmula deste capítulo, mas primeiro a enuncie. Os quadros chegam aleatoriamente a um canal de 100 Mbps para transmissão. Se estiver ocupado quando um quadro chegar, o canal aguardará sua vez em uma fila. O comprimento do quadro está distribuído exponencialmente com uma média de 10.000 bits/quadro. Para cada uma das taxas de chegada de quadros a seguir, determine o atraso experimentado pelo quadro médio, incluindo tanto o tempo de enfileiramento quanto o tempo de transmissão.
   (a) 90 quadros/s.
   (b) 900 quadros/s.
   (c) 9.000 quadros/s.
2. Um grupo de N estações compartilha um canal ALOHA original de 56 kbps. Cada estação transmite em média um quadro de 1.000 bits a cada 100 s, mesmo que o anterior ainda não tenha sido enviado (as estações podem, por exemplo, armazenar os quadros enviados em um buffer). Qual é o valor máximo de N?
3. Compare o atraso do ALOHA original com o do slotted ALOHA em carga baixa. Qual deles é menor? Explique sua resposta.
4. Uma grande população de usuários do ALOHA tenta gerar 50 solicitações/s, incluindo os quadros originais e as retransmissões. O tempo é dividido em unidades de 40 ms.
   (a) Qual é a chance de sucesso na primeira tentativa?

(b) Qual é a probabilidade de haver exatamente *k* colisões antes de obter sucesso?

(c) Qual é o número esperado de tentativas de transmissão necessárias?

5. Em um sistema slotted ALOHA com uma população infinita, o número médio de slots que uma estação aguarda entre uma colisão e sua retransmissão é 4. Represente em um diagrama a curva de variação do atraso com o throughput para esse sistema.

6. Qual é o tamanho de um slot de disputa em CSMA/CD para (a) um cabo twin de 2 Km (a velocidade de propagação do sinal é 82 por cemto da velocidade de propagação do sinal no vácuo)? e (b) um cabo de fibra óptica multimodo de 40 Km (a velocidade de propagação é 65 por cento da velocidade de propagação do sinal no vácuo)?

7. Quanto tempo uma estação *s* terá de esperar, na pior das hipóteses, antes de poder começar a transmitir seu quadro sobre uma LAN que use o protocolo bit-map básico?

8. No protocolo de contagem regressiva binária, explique como uma estação com número mais baixo pode ser impedida de enviar um pacote.

9. Dezesseis estações, numeradas de 1 a 16, estão disputando o uso de um canal compartilhado que emprega o protocolo tree walk adaptativo. Se todas as estações cujos endereços são números primos de repente ficarem disponíveis ao mesmo tempo, quantos slots de bits serão necessários para resolver a disputa?

10. Considere cinco estações sem fios, *A*, *B*, *C*, *D* e *E*. A estação *A* pode se comunicar com todas as outras estações. *B* pode se comunicar com *A*, *C* e *E*. *C* pode se comunicar com *A*, *B* e *D*. *D* pode se comunicar com *A*, *C* e *E*. *E* pode se comunicar com *A*, *D* e *B*.

    (a) Quando *A* está transmitindo para *B*, que outras comunicações são possíveis?

    (b) Quando *B* está transmitindo para *A*, que outras comunicações são possíveis?

    (c) Quando *B* está transmitindo para *C*, que outras comunicações são possíveis?

11. Seis estações, de *A* até *F*, se comunicam usando o protocolo MACA. Seria possível duas transmissões ocorrerem simultaneamente? Explique sua resposta.

12. Um prédio comercial de sete andares tem 15 escritórios adjacentes por andar. Cada escritório contém uma tomada (um soquete) para um terminal na parede frontal. Dessa forma, as tomadas formam uma grade retangular em um plano vertical, com uma distância de 4 m entre as tomadas, tanto no sentido horizontal quanto no vertical. Partindo do princípio de que é possível passar um cabo linear entre qualquer par de tomadas, seja no sentido horizontal, seja vertical ou diagonal, quantos metros de cabo seriam necessários para conectar todas as tomadas usando:

    (a) uma configuração em estrela com um único roteador no centro?

    (b) uma LAN 802.3 clássica?

13. Qual é a taxa baud da rede Ethernet clássica de 10 Mbps?

14. Estruture a codificação Manchester em uma Ethernet clássica para o fluxo de bits 0001110101.

15. Uma LAN CSMA/CD de 10 Mbps (não 802.3), com 1 Km de extensão, tem uma velocidade de propagação de 200 m/µs. Não são permitidos repetidores nesse sistema. Os quadros de dados têm 256 bits, incluindo 32 bits de cabeçalho, checksums e outras formas de overhead. O primeiro slot de bits depois de uma transmissão bem-sucedida é reservado para o receptor capturar o canal com o objetivo de enviar um quadro de confirmação de 32 bits. Qual será a taxa de dados efetiva, excluindo o overhead, se partirmos do princípio de que não há colisões?

16. Duas estações CSMA/CD estão tentando transmitir arquivos longos (de vários quadros). Depois que cada quadro é enviado, elas disputam o canal usando o algoritmo de backoff exponencial binário. Qual é a probabilidade de a disputa terminar na rodada de número *k*, e qual é o número médio de rodadas por período de disputa?

17. Um pacote IP a ser transmitido por uma rede Ethernet tem 60 bytes de comprimento, incluindo todos os seus cabeçalhos. Se o LLC não estiver em uso, será necessário utilizar preenchimento no quadro Ethernet? Em caso afirmativo, de quantos bytes?

18. Os quadros Ethernet devem ter pelo menos 64 bytes para garantir que o transmissor ainda estará ativo na eventualidade de ocorrer uma colisão na extremidade remota do cabo. O tamanho mínimo do quadro nas redes Fast Ethernet também é de 64 bytes, mas é capaz de transportar o mesmo número de bits com uma velocidade dez vezes maior. Como é possível manter o mesmo tamanho mínimo de quadro?

19. Alguns livros citam o tamanho máximo de um quadro Ethernet como 1.522 bytes em vez de 1.500 bytes. Eles estão errados? Explique sua resposta.

20. Quantos quadros por segundo a gigabit Ethernet pode manipular? Pense cuidadosamente e leve em conta todos os casos relevantes. *Dica*: o fato de ela ser uma gigabit Ethernet é importante.

21. Identifique duas redes que permitam que os quadros sejam reunidos em sequência. Por que é importante haver essa característica ?

22. Na Figura 4.24 são mostradas quatro estações, *A*, *B*, *C* e *D*. Qual das duas últimas estações você acha que está mais próxima de *A*, e por quê?

23. Dê um exemplo para mostrar que o RTC/CTS no protocolo 802.11 é um pouco diferente daquele do protocolo MACA.

24. Uma LAN sem fios com um PA tem dez estações clientes. Quatro estações possuem taxas de dados de 6 Mbps, quatro estações têm taxas de dados de 18 Mbps e as duas últimas estações têm taxas de dados de 54 Mbps. Qual é a taxa de dados experimentada por cada estação quando todas as dez estações estão transmitindo dados juntas e

(a) TXOP não é usada?

(b) TXOP é usada?

25. Suponha que uma LAN 802.11b de 11 Mbps esteja transmitindo quadros de 64 bytes em sequência por um canal de rádio com uma taxa de erros de bits igual a 10^{-7}. Quantos quadros por segundo serão danificados em média?

26. Uma rede 802.16 tem uma largura de canal de 20 MHz. Quantos bits/s podem ser enviados a uma estação do assinante?

27. Apresente duas razões pelas quais as redes poderiam usar um código de correção de erros em vez de detecção de erros e retransmissão.

28. Liste duas maneiras nas quais WiMAX é semelhante ao 802.11 e duas maneiras nas quais ele é diferente do 802.11.

29. Na Figura 4.31, observamos que um dispositivo Bluetooth pode estar em duas piconets ao mesmo tempo. Existe alguma razão pela qual um dispositivo não possa ser o mestre em ambas as piconets ao mesmo tempo?

30. Qual é o tamanho máximo do campo de dados para um quadro Bluetooth de 3 slots na taxa básica? Explique sua reposta.

31. A Figura 4.21 mostra diversos protocolos da camada física. Qual deles é o mais próximo do protocolo da camada física do Bluetooth? Qual é a maior diferença entre os dois?

32. Na Seção 4.6.6, mencionamos que a eficiência de um quadro de 1 slot com codificação de repetição é de cerca de 13 por cento na taxa de dados básica. Qual será a eficiência se, em vez disso, for usado um quadro de 5 slots com codificação de repetição na taxa de dados básica?

33. Os quadros de baliza na variante de espectro de dispersão de salto de frequência do 802.11 contêm o tempo de parada. Você acha que os quadros de baliza análogos no Bluetooth também contêm o tempo de parada? Explique sua resposta.

34. Suponha que haja dez etiquetas de RFID em torno de uma leitora de RFID. Qual é o melhor valor de Q? Qual é a probabilidade de que uma etiqueta responda sem colisão em determinado slot?

35. Liste algumas das questões de segurança de um sistema de RFID.

36. Um switch projetado para uso com Fast Ethernet tem uma placa interna que pode mover 10 Gbps. Na pior das hipóteses, quantos quadros/s ela pode tratar?

37. Descreva resumidamente a diferença entre os switches store-and-forward e cut-through.

38. Considere a LAN estendida conectada usando as bridges *B1* e *B2* na Figura 4.38(b). Suponha que as tabelas de hash nas duas bridges estejam vazias. Liste todas as portas em que um pacote será encaminhado para a seguinte sequência de transmissões de dados:

(a) *A* transmite um pacote para *C*.

(b) *E* transmite um pacote para *F*.

(c) *F* transmite um pacote para *E*.

(d) *G* transmite um pacote para *E*.

(e) *D* transmite um pacote para *A*.

(f) *B* transmite um pacote para *F*.

39. Os switches store-and-forward levam vantagem sobre os switches cut-through no que se refere a quadros danificados. Explique qual é essa vantagem.

40. Mencionamos, na Seção 4.8.3, que algumas bridges podem nem sequer estar presentes na spanning tree. Mostre um cenário no qual uma bridge pode não estar presente na spanning tree.

41. Para fazer as VLANs funcionarem, são necessárias tabelas de configuração nas bridges. E se as VLANs da Figura 4.44 usarem hubs em vez de switches? Os hubs também necessitam de tabelas de configuração? Por quê?

42. Na Figura 4.45, o switch no domínio final da tecnologia antiga do lado direito é um switch que reconhece VLANs. Seria possível usar ali um switch da tecnologia antiga? Nesse caso, como isso funcionaria? Se não, por que não?

43. Escreva um programa que simule o comportamento do protocolo CSMA/CD sobre Ethernet quando existirem *N* estações prontas para transmitir enquanto um quadro está sendo transmitido. Seu programa deve informar os momentos em que cada estação inicia com êxito a transmissão de seu quadro. Suponha que ocorra um pulso de clock em cada período de slot (51,2 μs) e que uma sequência de detecção de colisão e transmissão de interferência demore um período de slot. Todos os quadros têm o comprimento máximo permitido.

A camada de rede

Capítulo 5

A camada de rede está relacionada à transferência de pacotes da origem para o destino. Chegar ao destino pode exigir vários hops (saltos) em roteadores intermediários ao longo do percurso. Essa função contrasta claramente com a função da camada de enlace de dados, que tem o objetivo mais modesto de apenas mover quadros de uma extremidade de um fio para a outra. Portanto, a camada de rede é a camada mais baixa que lida com a transmissão ponto a ponto.

Para alcançar seus objetivos, a camada de rede deve conhecer a topologia da rede (ou seja, o conjunto de todos os roteadores e enlaces) e escolher os caminhos mais apropriados que a compõem, isso vale até mesmo para redes grandes. A camada de rede também deve ter o cuidado de escolher rotas que evitem sobrecarregar algumas das linhas de comunicação e roteadores enquanto deixa outras ociosas. Por fim, quando a origem e o destino estão em redes diferentes, ocorrem novos problemas, e cabe à camada de rede lidar com eles. Neste capítulo, estudaremos todas essas questões e as ilustraremos usando principalmente a Internet e o protocolo de sua camada de rede, o IP.

5.1 Questões de projeto da camada de rede

Nas seções a seguir, apresentaremos informações introdutórias sobre algumas das questões com as quais os projetistas da camada de rede devem se preocupar. Entre elas estão o serviço oferecido à camada de transporte e o projeto interno da rede.

5.1.1 Comutação de pacotes store-and-forward

Antes de começarmos a explicar os detalhes da camada de rede, vale a pena redefinir o contexto em que operam os protocolos da camada de rede. Esse contexto pode ser visto na Figura 5.1. Os principais componentes de rede são os equipamentos do ISP (roteadores conectados por linhas de transmissão), mostrados na elipse sombreada, e os equipamentos dos clientes, mostrados fora da elipse. O host *H1* está diretamente conectado a um dos roteadores do ISP, denominado *A*, talvez como um computador pessoal conectado a um modem DSL. Em contrapartida, *H2* está em uma LAN, que poderia ser a Ethernet de um escritório, com um roteador, *F*, pertencente e operado pelo cliente. Esse roteador também tem uma linha dedicada para o equipamento do ISP. Mostramos *F* fora da elipse porque ele não pertence ao ISP. Entretanto, para os propósitos deste capítulo, os roteadores nas instalações do cliente são considerados parte da rede do ISP, porque executam os mesmos algoritmos que os roteadores do ISP (e nossa principal preocupação aqui é o estudo dos algoritmos).

Esse equipamento é usado da maneira descrita a seguir. Um host com um pacote a enviar o transmite para o roteador mais próximo, seja em sua própria LAN, seja sobre um enlace ponto a ponto para o ISP. O pacote é armazenado ali até chegar completamente, de forma que o checksum possa ser conferido. Em seguida, ele é encaminhado para o próximo roteador ao longo do caminho, até alcançar o host de destino, onde ele é entregue. Esse mecanismo é a comutação de pacotes store-and-forward, como vimos em capítulos anteriores.

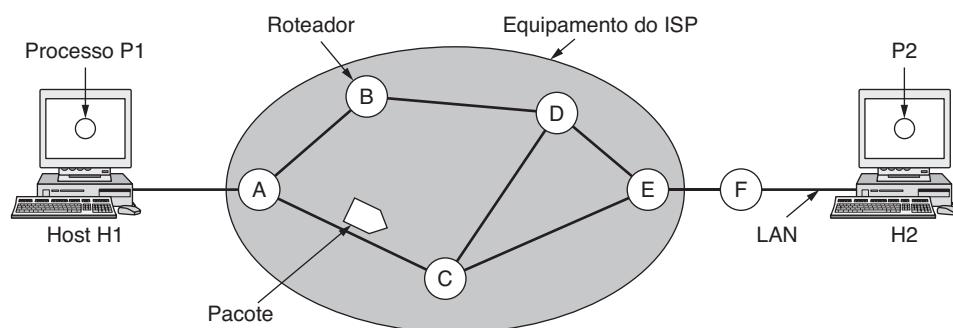



Figura 5.1 | O ambiente dos protocolos da camada de rede.

5.1.2 Serviços oferecidos à camada de transporte

A camada de rede oferece serviços à camada de transporte na interface entre as camadas de rede e de transporte. Uma questão importante é identificar os tipos de serviços que a camada de rede oferece à camada de transporte. Os serviços precisam ser cuidadosamente planejados tendo em vista os objetivos a seguir:

1. Os serviços devem ser independentes da tecnologia presente nos roteadores.
2. A camada de transporte deve ser isolada do número, do tipo e da topologia dos roteadores presentes.
3. Os endereços de rede que tornam os pacotes disponíveis para a camada de transporte devem usar um plano de numeração uniforme, mesmo nas LANs e WANs.

Tendo definido esses objetivos, os projetistas da camada de rede têm muita liberdade para escrever especificações detalhadas dos serviços a ser oferecidos à camada de transporte. Essa liberdade costuma se transformar em uma violenta batalha entre duas facções. A discussão se concentra na seguinte questão: a camada de rede deve fornecer serviço orientado a conexões ou não orientado a conexões?

Um lado, representado pela comunidade da Internet, alega que a tarefa dos roteadores é tão somente movimentar pacotes. Na visão dessas pessoas (baseada em quarenta anos de experiência com uma rede de computadores real), a rede é inerentemente não confiável, independentemente de como tenha sido projetada. Portanto, os hosts devem aceitar esse fato e fazer eles próprios o controle de erros (ou seja, detecção e correção de erros) e o controle de fluxo.

Esse ponto de vista leva à conclusão de que o serviço de rede deve ser não orientado a conexões, praticamente restrito às primitivas SEND PACKET e RECEIVE PACKET. Em particular, não devem ser feitos ordenação de pacotes nem controle de fluxo, pois os hosts cuidarão disso de qualquer maneira e, em geral, não há grande vantagem em fazer a mesma tarefa duas vezes. Esse raciocínio é um exemplo do **argumento fim a fim**, um princípio de projeto que tem sido muito influente na modelagem da Internet (Saltzer et al., 1984). Além disso, cada pacote deve ter o endereço de destino completo, pois todos são transportados independentemente de seus predecessores, se for o caso.

O outro lado, representado pelas companhias telefônicas, alega que a rede deve fornecer um serviço orientado a conexões confiável. Elas afirmam que os cem anos de experiência bem-sucedida com o sistema telefônico mundial servem como um excelente guia. De acordo com essa visão, a qualidade de serviço é o fator dominante e, sem conexões na rede, é muito difícil alcançar qualidade de serviço, em especial no caso de tráfego em tempo real, como voz e vídeo.

Mesmo depois de várias décadas, essa controvérsia ainda está muito viva. Antigamente, as redes de dados muito usadas, como a X.25 na década de 70 e sua sucessora, Frame Relay, na década de 80, eram orientadas à conexão. Contudo, desde os dias da ARPANET e da Internet inicial, as camadas de rede não orientadas à conexão cresceram muito em popularidade. O protocolo IP agora é um símbolo de sucesso sempre presente. Ele não recuou mesmo diante de uma tecnologia orientada à conexão, chamada ATM, que foi desenvolvida para aboli-lo na década de 80; em vez disso, agora é a ATM que tem seu uso em nichos e o IP que está assumindo as redes de telefonia. Debaixo dos panos, porém, a Internet está evoluindo e recursos orientados à conexão, como a qualidade de serviço, tornam-se mais importantes. Dois exemplos de tecnologias orientadas a conexões são MPLS (MultiProtocol Label Switching), que descreveremos neste capítulo, e VLANs, que vimos no Capítulo 4. As duas tecnologias são muito utilizadas.

5.1.3 Implementação do serviço não orientado a conexões

Depois de analisar as duas classes de serviço que a camada de rede pode oferecer a seus usuários, chegou a hora de vermos como essa camada funciona por dentro. São possíveis duas organizações, dependendo do tipo de serviço oferecido. Se for oferecido o serviço não orientado a conexões, os pacotes serão injetados individualmente na rede e roteados de modo independente uns dos outros. Não será necessária nenhuma configuração antecipada. Nesse contexto, os pacotes frequentemente são chamados **datagramas** (em analogia com os telegramas) e a rede será denominada **rede de datagramas**. Se for usado o serviço orientado a conexões, terá de ser estabelecido um caminho desde o roteador de origem até o de destino, antes de ser possível enviar quaisquer pacotes de dados. Essa conexão é chamada **circuito virtual**, em analogia aos circuitos físicos estabelecidos pelo sistema telefônico, e a rede é denominada **rede de circuitos virtuais**. Nesta seção, examinaremos as redes de datagramas; na próxima, estudaremos as redes de circuitos virtuais.

Vejamos agora como funciona uma rede de datagramas. Suponha que o processo *P1* da Figura 5.2 tenha uma mensagem longa para *P2*. Ele entrega a mensagem à camada de transporte, com instruções para que ela seja entregue a *P2* do host *H2*. O código da camada de transporte funciona em *H1*, em geral dentro do sistema operacional. Ele acrescenta um cabeçalho de transporte ao início da mensagem e entrega o resultado à camada de rede, que talvez simplesmente seja outro procedimento no sistema operacional.

Dest.	Tabela de A (inicial)	Tabela de A (depois)	Tabela de C	Tabela de E
A	–	–	A	C
B	B	B	A	D
C	C	C	–	C
D	B	B	E	D
E	C	B	E	–
F	C	B	E	F

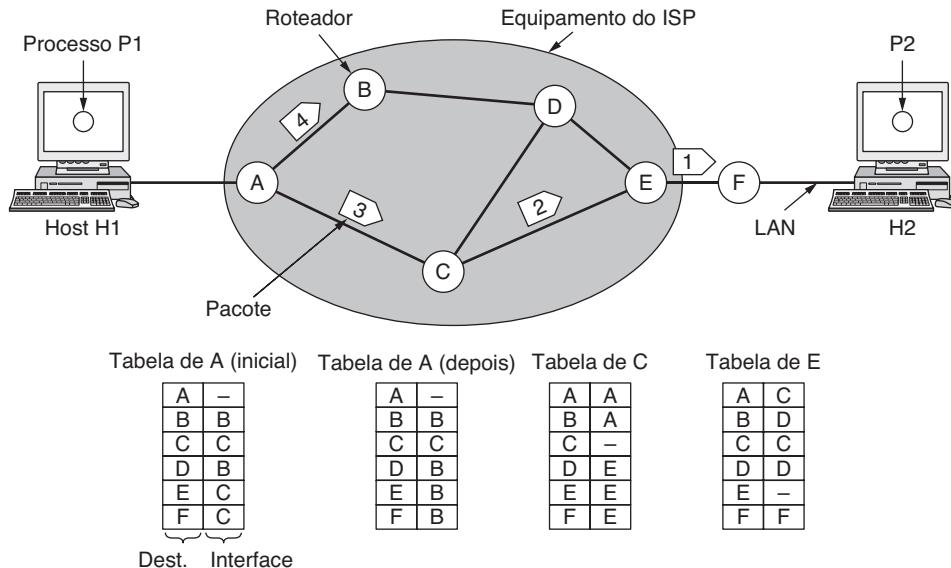



Figura 5.2 | Roteamento em uma rede de datagramas.

Vamos supor que a mensagem seja quatro vezes mais longa que o tamanho máximo de pacote e, portanto, que a camada de rede tenha de dividi-la em quatro pacotes, 1, 2, 3 e 4, e enviar cada um deles ao roteador *A*, usando algum protocolo ponto a ponto, como o PPP. Nesse ponto, o ISP assume o controle. Todo roteador tem uma tabela interna que informa para onde devem ser enviados os pacotes a ser entregues a cada possível destino. Cada entrada da tabela é um par que consiste em um destino e na interface de saída a ser utilizada para esse destino. Somente podem ser usadas interfaces diretamente conectadas. Por exemplo, na Figura 5.2, *A* tem apenas duas interfaces de saída — para *B* e *C* — e, assim, todo pacote recebido deve ser enviado a um desses roteadores, mesmo que o destino final seja algum outro roteador. A tabela de roteamento inicial de *A* é mostrada na figura sob o título 'inicial'.

Em *A*, os pacotes 1, 2 e 3 foram armazenados por algum tempo, tendo chegado ao enlace de entrada, e seus checksums conferidos. Em seguida, cada um deles foi encaminhado para *C*, de acordo com a tabela de *A*, dentro de um novo quadro. O pacote 1 foi então encaminhado para *E* e depois para *F*. Chegando a *F*, ele foi enviado dentro de um quadro para *H2* pela LAN. Os pacotes 2 e 3 seguem a mesma rota.

Entretanto, aconteceu algo diferente com o pacote 4. Quando chegou ao roteador *A*, ele foi enviado para o roteador *B*, embora seu destino também fosse *F*. Por alguma razão, *A* decidiu enviar o pacote 4 por uma rota diferente da que foi usada para os três primeiros pacotes. Talvez ele tenha tomado conhecimento de uma obstrução de tráfego em algum lugar ao longo do caminho *ACE* e tenha atualizado sua tabela de roteamento, como mostramos na figura sob o título 'depois'. O algoritmo que gerencia as tabelas e toma as decisões de roteamento é chamado **algoritmo de roteamento**. Estes constituem um dos principais assuntos que estudaremos neste capítulo. Como veremos, existem diferentes tipos de algoritmos.

O IP (Internet Protocol), que é a base para a Internet inteira, é um exemplo dominante de serviço de rede não orientado a conexões. Cada pacote transporta um endereço IP de destino que os roteadores utilizam para encaminhar cada pacote individualmente. Os endereços têm 32 bits nos pacotes IPv4 e 128 bits nos pacotes IPv6. Vamos descrever o IP com muito mais detalhes mais adiante neste capítulo.

5.1.4 Implementação do serviço orientado a conexões

No caso do serviço orientado a conexões, precisamos de uma rede de circuitos virtuais. Vejamos como ela funciona. A ideia que rege os circuitos virtuais é evitar a necessidade de escolher uma nova rota para cada pacote enviado, como na Figura 5.2. Em vez disso, quando uma conexão é estabelecida, escolhe-se uma rota desde a máquina de origem até a máquina de destino, como parte da configuração da conexão, e essa rota é armazenada em tabelas internas dos roteadores. Essa rota é usada por todo o tráfego que flui pela conexão, exatamente como ocorre no sistema telefônico. Quando a conexão é liberada, o circuito virtual também é encerrado. Com o serviço orientado a conexões, cada pacote transporta um identificador, informando a qual circuito virtual ele pertence.

Como exemplo, considere a situação da Figura 5.3. Na figura, o host *H1* estabeleceu a conexão 1 com *H2*. Ela é memorizada como a primeira entrada de cada uma das tabelas de roteamento. A primeira linha da tabela de *A* infor-

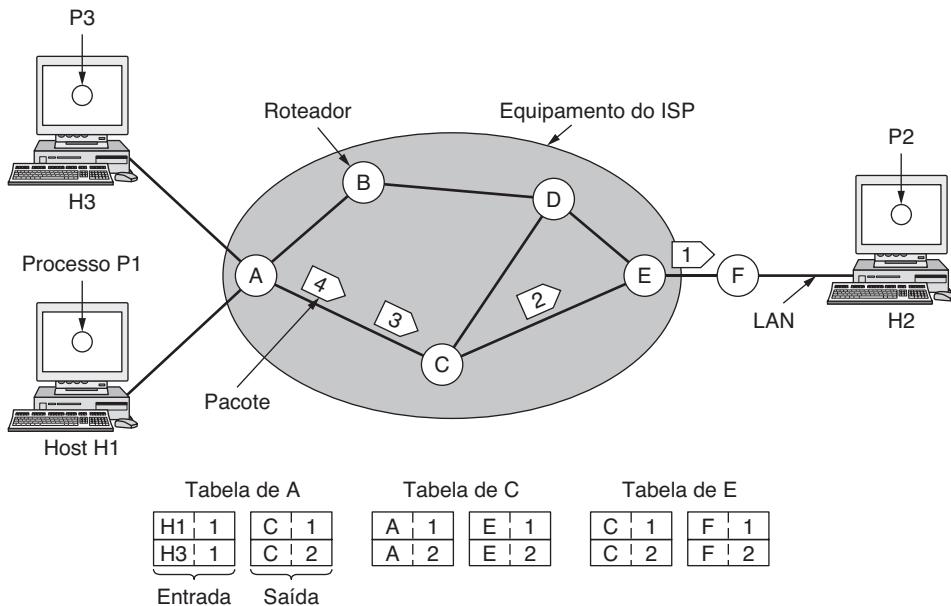



Figura 5.3 | Roteamento em uma rede de circuitos virtuais.

ma que, se um pacote contendo o identificador de conexão 1 chegar de *H1*, ele será enviado ao roteador *C* e receberá o identificador de conexão 1. De modo semelhante, a primeira entrada em *C* faz o roteamento do pacote para *E*, também com o identificador de conexão 1.

Agora, vejamos o que acontece se *H3* também quiser estabelecer uma conexão com *H2*. Ele escolhe o identificador de conexão 1 (porque está iniciando a conexão, e essa é sua única conexão) e informa à rede que ela deve estabelecer o circuito virtual. Isso conduz à segunda linha nas tabelas. Observe que, nesse caso, temos um conflito, pois, embora *A* possa distinguir facilmente os pacotes da conexão 1 provenientes de *H1* dos pacotes da conexão 1 que vêm de *H3*, *C* não tem como fazer o mesmo. Por essa razão, *A* atribui um identificador de conexão diferente ao tráfego de saída correspondente à segunda conexão. Evitar conflitos desse tipo é a razão pela qual os roteadores precisam ter a capacidade de substituir identificadores de conexões para os pacotes de saída.

Em alguns contextos, essa operação é chamada **troca de rótulos**. Um exemplo de serviço de rede orientado a conexões é o **MPLS** (**MultiProtocol Label Switching**). Ele é usado dentro das redes do ISP na Internet, com os pacotes IP inseridos em um cabeçalho MPLS com um identificador ou rótulo de conexão de 20 bits. O MPLS normalmente fica oculto aos clientes, com o ISP estabelecendo conexões a longas distâncias para grandes volumes de tráfego, mas ele está sendo cada vez mais utilizado para ajudar na qualidade do serviço e também em outras tarefas de gerenciamento do tráfego do ISP. Veremos mais a respeito do MPLS em outro ponto deste capítulo.

5.1.5 Comparação entre redes de circuitos virtuais e de datagramas

Tanto os circuitos virtuais como os datagramas têm seus fãs e seus detratores. Agora, vamos tentar resumir os argumentos de ambos os lados. As principais questões estão listadas na Tabela 5.1, ainda que os puristas provavelmente venham a encontrar um exemplo contrário para tudo o que for apresentado na figura.

Dentro da rede, existem vários dilemas entre circuitos virtuais e datagramas. Um deles é o compromisso entre o tempo de configuração e o tempo de análise do endereço. O uso de circuitos virtuais exige uma fase de configuração, que leva tempo e consome recursos. Contudo, quando esse preço é pago, é fácil descobrir o que fazer com um pacote de dados em uma rede de circuitos virtuais: o roteador simplesmente usa o número de circuito para indexar sua tabela e descobrir para onde vai o pacote. Em uma rede de datagramas, nenhuma configuração é necessária, mas é necessário um procedimento de pesquisa mais complicado para mapear o endereço de destino.

Uma questão relacionada é que os endereços de destino usados nas redes de datagrama são maiores que os números de circuito usados nas redes de circuito virtual, pois eles têm um significado global. Se os pacotes tenderem a ser muito pequenos, incluir um endereço de destino completo em cada pacote poderá representar um volume significativo de overhead e, portanto, haverá desperdício de largura de banda.

Outra questão é a quantidade de espaço exigido em tabelas na memória do roteador. Uma rede de datagramas precisa ter uma entrada para cada destino possível, en-

Questão	Rede de datagramas	Rede de circuitos virtuais
Configuração de circuitos	Desnecessária	Obrigatória
Endereçamento	Cada pacote contém os endereços completos de origem e de destino	Cada pacote contém um pequeno número do circuito virtual
Informações sobre o estado	Os roteadores não armazenam informações sobre o estado das conexões	Cada circuito virtual requer espaço em tabelas de roteadores por conexão
Roteamento	Cada pacote é roteado independentemente	A rota é escolhida quando o circuito virtual é estabelecido; todos os pacotes seguem essa rota
Efeito de falhas no roteador	Nenhum, com exceção dos pacotes perdidos durante a falha	Todos os circuitos virtuais que tiverem passado pelo roteador que apresentou o defeito serão encerrados
Qualidade de serviço	Difícil	Fácil, se for possível alocar recursos suficientes com antecedência para cada circuito virtual
Controle de congestionamento	Difícil	Fácil, se for possível alocar recursos suficientes com antecedência para cada circuito virtual

Tabela 5.1 | Comparação entre redes de circuitos virtuais e de datagramas.

quanto uma rede de circuitos virtuais só precisa de uma entrada para cada circuito virtual. No entanto, essa vantagem é um pouco ilusória, pois pacotes de configuração de conexões também têm de ser roteados e, da mesma forma que os datagramas, eles usam endereços de destino.

Os circuitos virtuais têm algumas vantagens na garantia da qualidade de serviço e ao evitar o congestionamento dentro da rede, pois os recursos (por exemplo, buffers, largura de banda e ciclos de CPU) podem ser reservados antecipadamente, quando a conexão é estabelecida. Quando os pacotes começarem a chegar, a largura de banda e a capacidade do roteador necessárias já estarão instaladas. Com uma rede de datagramas, é mais difícil evitar o congestionamento.

No caso de sistemas de processamento de transações (por exemplo, lojas que utilizam o telefone para verificar compras com cartões de crédito), o overhead necessário para configurar e limpar um circuito virtual pode reduzir facilmente o uso do circuito. Caso se espere que a maior parte do tráfego seja desse tipo, o uso de circuitos virtuais esporádicos dentro da rede fará pouco sentido. Por outro lado, circuitos virtuais permanentes, configurados manualmente e que duram meses ou anos, talvez sejam úteis nessa situação.

Os circuitos virtuais também têm um problema de vulnerabilidade. Se um roteador apresentar uma falha e perder sua capacidade de memória, mesmo que volte um segundo depois, todos os circuitos virtuais que estiverem passando por ele terão de ser interrompidos. Por outro lado, se um roteador de datagramas ficar fora do ar, somente serão afetados os usuários cujos pacotes estiverem enfileirados no roteador naquele momento (e talvez nem todos eles, pois o transmissor provavelmente os retransmitirá em breve). A perda de uma linha de comunicação é fatal para os circuitos virtuais que a utilizam, mas pode ser compensada com facilidade se forem usados datagramas. Estes também permitem que os roteadores equilibrem o tráfego pela rede, pois as rotas podem ser parcialmente alteradas durante uma longa sequência de transmissões de pacotes.

5.2 Algoritmos de roteamento

A principal função da camada de rede é rotear pacotes da máquina de origem para a máquina de destino. Na maioria das redes, os pacotes necessitarão de vários hops para cumprir o trajeto. A única exceção importante diz respeito às redes de broadcast, mas mesmo aqui o roteamento depende do fato de a origem e o destino não estarem na mesma rede. Os algoritmos que escolhem as rotas e as estruturas de dados que elas utilizam constituem um dos elementos mais importantes do projeto da camada de rede.

O **algoritmo de roteamento** é a parte do software da camada de rede responsável pela decisão sobre a interface de saída a ser usada na transmissão do pacote de entrada. Se a rede internamente utilizar datagramas, essa decisão deverá ser tomada mais de uma vez para cada pacote de dados recebido, pois a melhor rota pode ter sido alterada desde a última vez. Se a rede internamente utilizar circuitos virtuais, as decisões de roteamento serão tomadas somente quando um novo circuito virtual estiver sendo estabelecido. Daí em diante, os pacotes de dados seguirão a rota previamente estabelecida. Às vezes, essa última circunstância é chamada **roteamento por sessão**, pois uma rota permanece em vigor durante toda uma sessão de comunicação com o usuário (por exemplo, uma sessão de login em uma rede VPN).

Algumas vezes, é útil fazer distinção entre o roteamento, que é a tomada de decisão sobre quais rotas utilizar, e o encaminhamento, que acontece quando um pacote chega. Podemos imaginar que um roteador tem dois processos internamente. Um deles trata cada pacote que chega, procurando a interface de saída que será usada em sua tabela de roteamento. Esse processo é o **encaminhamento**. O outro processo é responsável pelo preenchimento e pela atualização das tabelas de roteamento. É nesse processo que o algoritmo de roteamento entra em cena.

Mesmo que as rotas sejam escolhidas independentemente para cada pacote ou apenas quando novas conexões são estabelecidas, certas propriedades são desejáveis em um algoritmo de roteamento: exatidão, simplicidade, robustez, estabilidade, equidade e eficiência. Os itens exatidão e simplicidade são autoexplicativos, mas, em princípio, talvez a necessidade de robustez seja menos óbvia. Uma vez que uma rede de maior porte é instalada, espera-se que ela funcione continuamente durante anos sem apresentar nenhuma falha no sistema. Durante esse período, haverá todos os tipos de falhas de hardware e software. Os hosts, os roteadores e as interfaces falharão repetidamente, e a topologia mudará muitas vezes. O algoritmo de roteamento deve ser capaz de aceitar as alterações de topologia e de tráfego sem exigir que todas as tarefas de todos os hosts sejam interrompidas e que a rede seja reiniciada sempre que algum roteador apresentar falha.

A estabilidade também é um objetivo importante do algoritmo de roteamento. Existem algoritmos que nunca convergem para um conjunto viável de rotas, independentemente do tempo em que são executados. Um algoritmo estável alcança um ponto de equilíbrio e permanece nesse estado. Ele também deve convergir rapidamente, pois a comunicação pode ser interrompida até que o algoritmo de roteamento tenha alcançado um ponto de equilíbrio.

A equidade e a eficiência podem parecer óbvias, pois certamente ninguém faria oposição a elas. No entanto, como se vê, com frequência elas têm objetivos conflitantes. Como um exemplo simples desse conflito, observe a Figura 5.4. Suponha que o tráfego entre *A* e *A*′, entre *B* e *B*′ e entre *C* e *C*′ seja suficiente para saturar os enlaces horizontais. Para maximizar o fluxo total, o tráfego de *X* para *X*′ deve ser desativado por completo. Infelizmente, talvez *X* e *X*′ não vejam a situação dessa maneira. É evidente que se faz necessário um meio-termo entre eficiência global e equidade para as conexões individuais.

Antes de tentarmos encontrar um meio-termo entre equidade e otimização, devemos decidir o que estamos buscando otimizar. A minimização do atraso médio do pacote é uma candidata óbvia, e o mesmo vale para a maximização do throughput total da rede. Além disso, esses dois objetivos também estão em conflito, pois operar qualquer sistema de enfileiramento em uma velocidade próxima a sua capacidade máxima implica um longo atraso de enfileiramento. Como meio-termo, muitas redes tentam minimizar a distância que um pacote deve percorrer, ou simplesmente reduzir o número de hops que um pacote deve percorrer. Qualquer uma dessas escolhas tende a melhorar o atraso e também reduzir a largura de banda consumida por volume de pacotes, o que, por sua vez, também tende a melhorar o throughput.

Os algoritmos de roteamento podem ser agrupados em duas classes principais: não adaptativos e adaptativos. Os **algoritmos não adaptativos** não baseiam suas decisões de roteamento em medidas ou estimativas do tráfego e da topologia atuais. Em vez disso, a escolha da rota a ser utilizada para ir de *I* até *J* (para todo *I* e todo *J*) é previamente calculada off-line, sendo transferida para os roteadores quando a rede é iniciada. Às vezes esse procedimento é chamado **roteamento estático**. Por não responder bem a falhas, o roteamento estático é mais útil para situações em que a escolha de rotas é óbvia. Por exemplo, o roteador *F* na Figura 5.3 deve enviar para o roteador *E* os pacotes direcionados à rede, independentemente do destino final.

Em contraste, os **algoritmos adaptativos** alteram as decisões de roteamento para refletir mudanças na topologia e, normalmente, também no tráfego. Esses algoritmos de **roteamento dinâmico** diferem em termos do lugar em que obtêm suas informações (por exemplo, no local, de roteadores adjacentes ou de todos os roteadores), do momento em que alteram as rotas (por exemplo, quando a topologia muda ou a cada ΔT segundos, quando a carga se

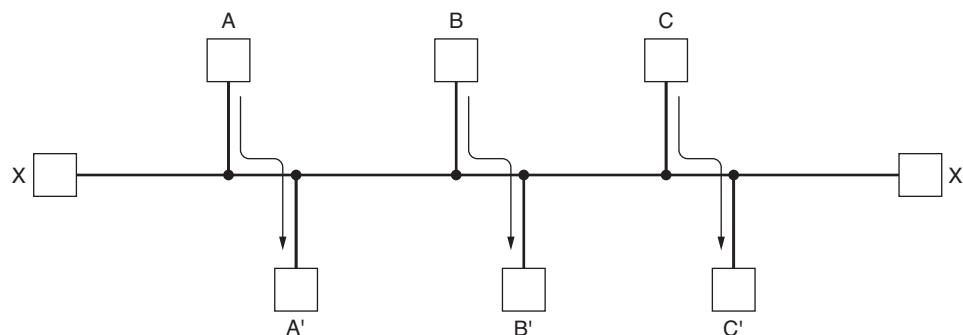



Figura 5.4 Rede com conflito entre equidade e eficiência.

altera) e da métrica utilizada na otimização (por exemplo, distância, número de hops ou tempo estimado de tráfego).

Nas próximas seções, trataremos de uma variedade de algoritmos de roteamento. Os algoritmos abordam modelos de entrega além de transmitir um pacote de uma origem para um destino. Às vezes o objetivo é enviar o pacote para vários, todos ou um entre um conjunto de destinos. Todos os algoritmos de roteamento que descrevemos aqui tomam decisões com base na topologia; deixamos a possibilidade de decisões com base nos níveis de tráfego para a Seção 5.3.

5.2.1 O princípio de otimização

Antes de estudarmos algoritmos específicos, talvez valha a pena lembrar que é possível criar uma descrição geral das rotas ideais sem levar em conta a topologia ou o tráfego de rede. Essa descrição é conhecida como **princípio de otimização** (Bellman, 1957). Esse princípio estabelece que, se o roteador *J* estiver no caminho ideal entre o roteador *I* e o roteador *K*, o caminho ideal de *J* até *K* também estará na mesma rota. Para confirmar isso, chame a parte da rota entre *I* e *J* de r_1 e o restante de r_2. Se existisse uma rota melhor que r_2 entre *J* e *K*, ela poderia ser concatenada com r_1 para melhorar a rota entre *I* e *K*, contradizendo nossa afirmação de que r_1r_2 é ideal.

Como consequência direta do princípio de otimização, podemos observar que o conjunto de rotas ideais de todas as origens para determinado destino forma uma árvore com raiz no destino. Uma árvore como essa é chamada **árvore de escoamento** e está ilustrada na Figura 5.5(b), em que a métrica de distância é o número de hops. O objetivo de todos os algoritmos de roteamento é descobrir e utilizar as árvores de escoamento em todos os roteadores.

Observe que uma árvore de escoamento não é necessariamente exclusiva; pode haver outras árvores com os mesmos tamanhos de caminho. Se permitirmos que todos os caminhos possíveis sejam escolhidos, a árvore se torna uma estrutura mais geral chamada **DAG** (**Directed Acyclic Graph**). Os DAGs não possuem loops. Usaremos árvores de escoamento como uma abreviação conveniente para esses dois casos. Ambos os casos também dependem da suposição técnica de que os caminhos não interferem uns nos outros. Assim, por exemplo, um engarrafamento no trânsito em um caminho não causará o desvio para outro caminho.

Como uma árvore de escoamento é de fato uma árvore, ela não contém loops; portanto, cada pacote será entregue dentro de um número finito e limitado de hops. Na prática, nem tudo é tão fácil assim. Enlaces e roteadores podem sair do ar e voltar à atividade durante a operação; desse modo, diferentes roteadores podem ter estimativas distintas sobre a topologia atual. Além disso, empregamos alguns artifícios para resolver a seguinte questão: cada roteador deve obter individualmente as informações sobre a base de cálculo de sua árvore de escoamento ou esses dados serão obtidos por algum outro meio? Retornaremos a essa questão em breve. Contudo, o princípio da otimização e a árvore de escoamento permitem que se faça um teste de benchmark para detectar que outros algoritmos de roteamento podem ser medidos.

5.2.2 Roteamento pelo caminho mais curto

Iniciaremos nosso estudo prático de algoritmos de roteamento com uma técnica simples para calcular os caminhos ideais a partir de uma imagem completa da rede. Esses caminhos são aqueles que queremos que um algoritmo de roteamento distribuído encontre, embora nem todos os roteadores possam conhecer todos os detalhes da rede.

A ideia é criar um grafo da rede, com cada nó do grafo representando um roteador e cada arco indicando uma interface de comunicação, ou enlace. Para escolher uma rota entre determinado par de roteadores, o algoritmo simplesmente encontra o caminho mais curto entre eles no grafo.

O conceito de **caminho mais curto** merece uma explicação. Uma forma de medir o comprimento do caminho é usar o número de hops. Empregando-se essa métrica, os caminhos *ABC* e *ABE* da Figura 5.6 são igualmente longos. Outra métrica é a distância geográfica em quilômetros e, nesse caso, *ABC* é claramente muito mais longo que *ABE* (supondo-se que a figura tenha sido desenhada em escala).

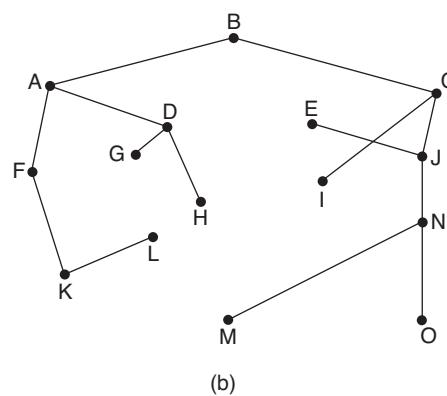
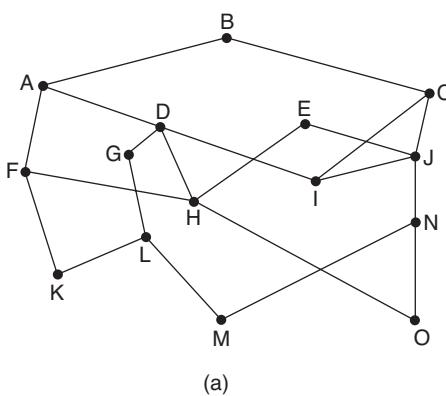



Figura 5.5 (a) Uma rede. (b) Uma árvore de escoamento para o roteador *B*.

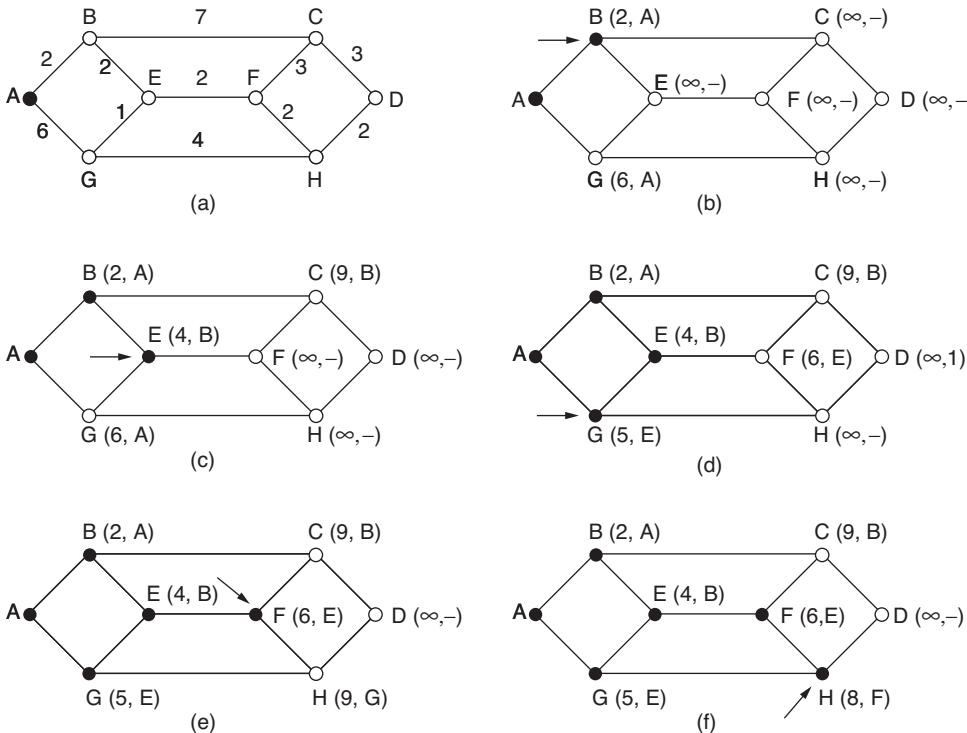



Figura 5.6 | As seis primeiras etapas utilizadas no cálculo do caminho mais curto de *A* até *D*. As setas indicam o nó ativo.

Entretanto, muitas outras métricas também são possíveis além do número de hops e da distância física. Por exemplo, cada arco poderia ser identificado com o atraso médio de enfileiramento e de transmissão referente a um pacote de teste padrão, de acordo com as especificações de testes executados a cada hora. Nesse grafo, o caminho mais curto é o caminho mais rápido, e não o caminho com menor número de arcos ou quilômetros.

No caso geral, os rótulos dos arcos podem ser calculados como uma função da distância, da largura de banda, do tráfego médio, do custo de comunicação, do atraso medido e de outros fatores. Alterando-se a função de ponderação (atribuição de pesos), o algoritmo então calcularia o caminho 'mais curto' medido de acordo com qualquer critério ou como uma combinação de critérios.

São conhecidos diversos algoritmos para calcular o caminho mais curto entre dois nós de um grafo. O algoritmo que vamos examinar agora se deve a Dijkstra (1959) e encontra os caminhos mais curtos entre uma origem e todos os destinos na rede. Cada nó é identificado (entre parênteses) por sua distância a partir do nó de origem ao longo do melhor caminho conhecido. As distâncias devem ser não negativas, como acontecerá se elas forem baseadas em quantidades reais, como largura de banda e atraso. Inicialmente, nenhum caminho é conhecido; portanto, todos os nós são rotulados com infinito. À medida que o algoritmo prossegue e os caminhos são encontrados, os rótulos podem mudar, refletindo melhores caminhos. Um rótulo pode ser provisório ou permanente. No início, todos são provisórios. Quando se descobre que um rótulo representa o caminho mais curto possível até a origem desse nó, ele se torna permanente e, daí em diante, nunca mais é alterado.

Para ilustrar como funciona o algoritmo de identificação, vamos examinar o grafo ponderado não orientado mostrado na Figura 5.6(a), em que os pesos representam, por exemplo, a distância. Desejamos encontrar o caminho mais curto de *A* até *D*. Começamos marcando o nó *A* como permanente, o que é indicado por um círculo preenchido. A seguir, examinamos separadamente cada um dos nós adjacentes a *A* (o nó ativo), alterando o rótulo de cada um deles para indicar a distância até *A*. Sempre que um nó é rotulado novamente, ele também é rotulado com o nó a partir do qual o teste foi feito; assim, podemos reconstruir o caminho final mais tarde. Se a rede tivesse mais de um caminho mais curto de *A* até *D* e quiséssemos encontrar todos eles, precisaríamos nos lembrar de todos os nós de teste que poderiam alcançar um nó com a mesma distância.

Após examinarmos cada um dos nós adjacentes a *A*, verificamos todos os nós provisoriamente rotulados no grafo inteiro e tornamos permanente o nó que tem o menor rótulo, como mostra a Figura 5.6(b). Esse nó passa a ser o novo nó ativo.

Agora, começamos por *B* e examinamos todos os nós adjacentes a ele. Se a soma do rótulo de *B* e a distância entre *B* e o nó que está sendo considerado for menor que o

rótulo desse nó, teremos um caminho mais curto; portanto, o nó será rotulado novamente.

Depois que todos os nós adjacentes ao nó ativo tiverem sido inspecionados e os rótulos provisórios tiverem sido alterados, na medida do possível, o grafo inteiro será pesquisado até ser encontrado o nó com o rótulo provisório de menor valor. Esse nó passará a ser o nó permanente e se tornará o nó ativo na próxima iteração. A Figura 5.6 mostra as seis primeiras etapas do algoritmo.

Para saber por que o algoritmo funciona, observe a Figura 5.6(c). Nesse momento, tornamos *E* permanente. Suponha a existência de um caminho mais curto que *ABE*, digamos, *AXYZE* (para algum X e Y). Há duas possibilidades: ou o nó *Z* já se tornou permanente ou não. Se ele já se tornou permanente, então *E* já foi testado (na iteração que se segue àquela em que *Z* se tornou permanente); assim, o caminho *AXYZE* não escapou à nossa atenção e, portanto, não pode ser um caminho mais curto.

Agora, leve em conta a hipótese de *Z* ainda ter um rótulo provisório. Então, o rótulo *Z* é maior ou igual ao de *E* e, nesse caso, *AXYZE* não pode ser um caminho mais curto que *ABE*. Se o rótulo for menor que o de *E*, então *Z* e não *E* se tornará permanente primeiro, permitindo que *E* seja testado a partir de *Z*.

Esse algoritmo é mostrado no Quadro 5.1. As variáveis globais *n* e *dist* descrevem o grafo e são inicializadas antes de *shortest_path* ser chamado. A única diferença entre o programa e o algoritmo descrito anteriormente é que, no Quadro 5.1, calculamos o caminho mais curto a partir do nó terminal *t*, em vez de começarmos no nó de origem, *s*.

Como os caminhos mais curtos de *t* até *s* em um grafo não orientado são iguais ao caminho mais curto de *s* até *t*, não importa em que extremidade comecemos. A razão para a pesquisa no sentido inverso é que cada nó é rotulado com seu predecessor em vez de ser rotulado com seu sucessor. Quando o caminho final for copiado na variável de saída, *path*, o caminho será então invertido. Os dois efeitos reversos se cancelarão e a resposta será produzida na ordem correta.

5.2.3 Flooding

Quando o algoritmo de roteamento é implementado, cada roteador precisa tomar decisões com base no conhecimento local, não na imagem completa da rede. Uma técnica local simples é a de **flooding** (inundação), na qual cada pacote de entrada é enviado para cada interface de saída, exceto para aquela em que chegou.

Evidentemente, o algoritmo de inundação gera uma vasta quantidade de pacotes duplicados, na verdade um número infinito, a menos que algumas medidas sejam tomadas para tornar o processo mais lento. Uma dessas medidas é ter um contador de hops contido no cabeçalho de cada pacote; o contador é decrementado em cada hop, com o pacote sendo descartado quando o contador atingir zero. O ideal é que o contador de hops seja iniciado com a distância do caminho desde a origem até o destino. Se não souber o tamanho do caminho, o transmissor poderá iniciar o contador com o valor referente ao pior caso, ou seja, o diâmetro total da rede.

O flooding com um contador de hops pode produzir um número exponencial de pacotes duplicados à medida que o contador de saltos ou hops duplica os pacotes já vistos. Um modo melhor para conter o processo de flooding é controlar quais pacotes foram transmitidos por essa técnica, a fim de evitar transmiti-los uma segunda vez. Uma forma de conseguir isso é fazer o roteador de origem inserir um número de sequência em cada pacote recebido de seus hosts. Portanto, cada roteador precisará de uma lista por roteador de origem informando quais números de sequência originários desse ponto já foram vistos. Se houver um pacote de entrada na lista, ele não será transmitido por flooding.

Para evitar que as listas cresçam indefinidamente, cada uma delas deve ser incrementada de acordo com um contador *k*, o que significa que todos os números de sequência até *k* foram vistos. Quando um pacote for recebido, será fácil verificar se ele é uma cópia (comparando seu número de sequência com *k*); se for, ele será descartado. Além disso, a lista completa abaixo de *k* não é necessária, visto que *k* resume essa lista.

O algoritmo de flooding não é prático para enviar a maioria dos pacotes, mas tem sua utilidade. Primeiro, ele garante que um pacote seja entregue a cada nó na rede. Isso pode ser um desperdício se houver um único destino precisando do pacote, mas é eficiente para a difusão de informações. Em redes sem fios, todas as mensagens transmitidas por uma estação podem ser recebidas por todas as outras estações dentro de seu alcance de rádio, o que, na verdade, representa o flooding, e alguns algoritmos empregam essa propriedade.

Em segundo lugar, o flooding é tremendamente robusto. Mesmo que grandes quantidades de roteadores estejam sobrecarregadas de bits (por exemplo, em uma rede militar localizada em uma zona de guerra), o flooding encontrará um caminho, se existir, para levar um pacote ao seu destino. O flooding também exige pouco no modo de configuração. Os roteadores só precisam conhecer seus vizinhos. Isso significa que o flooding pode ser usado como um bloco de montagem para outros algoritmos de roteamento que são mais eficientes, porém precisam de mais configuração. O flooding também pode ser usado como uma unidade de medida que servirá como base de comparação com outros algoritmos de roteamento. O flooding sempre escolhe o caminho mais curto, pois todos os caminhos possíveis são selecionados em paralelo. Em consequência disso, nenhum outro algoritmo é capaz de produzir um atraso de menor duração (se ignorarmos o overhead gerado pelo próprio processo de inundação).

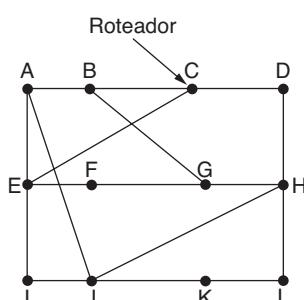
```
#define MAX_NODES 1024                          /* número máximo de nós */
#define INFINITY 1000000000                     /* um número maior que cada caminho máximo */
int n, dist[MAX_NODES][MAX_NODES];              /* dist[i][j] é a distância de i até j */

void shortest_path(int s, int t, int path[])
{ struct state{                                 /* o caminho sendo usado */
      int predecessor;                          /* nó anterior */
      int length;                               /* distância da origem até este nó */
      enum {permanent, tentative} label;        /* estado do rótulo */
  } state[MAX_NODES];

  int i, k, min;
  struct state *p;

  for (p = &state[0]; p < &state[n]; p++){      /*inicializa estado */
      p->predecessor = −1;
      p->length = INFINITY;
      p->label = tentative;
  }
  state[t].length = 0; state[t].label = permanent;
  k = t;                                        /* k é o nó ativo inicial */
  do {                                          /* Existe caminho melhor a partir de k? */
      for (i = 0; i < n; i++)                   /* este grafo tem n nós */
          if (dist[k][i] != 0 && state[i].label == tentative) {
              if (state[k].length + dist[k][i] < state[i].length) {
                  state[i].predecessor = k;
                  state[i].length = state[k].length + dist[k][i];
              }
          }
      /* Acha o nó provisoriamente rotulado com o menor rótulo. */
      k = 0; min = INFINITY;
      for (i = 0; i < n; i++)
          if (state[i].label == tentative && state[i].length < min) {
              min = state[i].length;
              k = i;
          }
      state[k].label = permanent;
  } while (k != s);

  /* Copia o caminho para array de saída. */
  i = 0; k = s;
  do {path[i++] = k; k = state[k].predecessor; } while (k >= 0);
}
```

Quadro 5.1 | O algoritmo de Dijkstra para calcular o caminho mais curto através de um grafo.

5.2.4 Roteamento por vetor de distância

Geralmente, as modernas redes de computadores utilizam algoritmos de roteamento dinâmicos que são mais complexos que o flooding, porém mais eficientes, porque encontram os caminhos mais curtos para a topologia atual. Dois algoritmos dinâmicos específicos, o roteamento por vetor de distância e o roteamento de estado de enlace, são os mais conhecidos. Nesta seção, vamos estudar o primeiro desses algoritmos. Na próxima seção, estudaremos o segundo.

Os algoritmos de **roteamento por vetor de distância** operam fazendo cada roteador manter uma tabela (isto é, um vetor) que fornece a melhor distância conhecida até cada destino e determina qual enlace deve ser utilizado para chegar lá. Essas tabelas são atualizadas por meio da troca de informações com os vizinhos. No fim, cada roteador saberá o melhor enlace para alcançar cada destino.

Às vezes, o algoritmo de roteamento por vetor de distância recebe outros nomes, sendo o mais comum o algoritmo de roteamento distribuído de **Bellman-Ford**, que recebeu o nome dos pesquisadores que o desenvolveram (Bellman, 1957; e Ford e Fulkerson, 1962). Ele foi o algoritmo de roteamento original da ARPANET e também foi utilizado na Internet, com o nome RIP.

No roteamento por vetor de distância, cada roteador mantém uma tabela de roteamento indexada para cada roteador da rede e que contém uma entrada para cada um deles. Essa entrada contém duas partes: a interface de saída preferencial a ser utilizada para o destino e uma estimativa da distância até esse destino. A métrica utilizada pode ser o número de hops ou outra medida, conforme discutimos para o cálculo do caminho mais curto.

Presume-se que o roteador conheça a 'distância' até cada um de seus vizinhos. Se a métrica for o hop, a distância será de apenas um hop. Se for o atraso de propagação, o roteador poderá medi-lo diretamente com pacotes ECHO/REPLY especiais, que o receptor identifica com um registro de tempo e transmite de volta o mais rápido que puder.

Por exemplo, suponha que o atraso seja usado como métrica e que o roteador saiba qual é o atraso até cada um de seus vizinhos. Uma vez a cada *T* ms, cada roteador envia a cada vizinho uma lista de seus atrasos estimados até cada destino. Ele também recebe uma lista semelhante de cada vizinho. Imagine que uma dessas tabelas tenha acabado de chegar do vizinho *X*, sendo X_i a estimativa de *X* sobre o tempo que ele levará para chegar até o roteador *i*. Se o roteador souber que o atraso para *X* é de *m* ms, ele também saberá que pode alcançar o roteador *i* por meio de *X* em $X_i + m$ ms. Efetuando esse cálculo para cada vizinho, um roteador pode descobrir qual estimativa parece ser a melhor, e também pode usar essa estimativa e o enlace correspondente em sua nova tabela de roteamento. Observe que a antiga tabela de roteamento não é utilizada no cálculo.

Esse processo de atualização é ilustrado na Figura 5.7. A parte (a) mostra uma rede. As quatro primeiras colunas da parte (b) mostram os vetores de atraso recebidos dos vizinhos do roteador *J*. *A* alega ter um atraso de 12 ms até *B*, um atraso de 25 ms até *C*, um atraso de 40 ms até *D* etc. Suponha que *J* tenha medido ou estimado seu atraso até seus vizinhos *A*, *I*, *H* e *K* como 8, 10, 12 e 6 ms, respectivamente.

Considere a forma como *J* calcula sua nova rota até o roteador *G*. Ele sabe que pode chegar até *A* em 8 ms e *A* alega ser capaz de chegar a *G* em 18 ms; portanto, *J* sabe que pode contar com um atraso de 26 ms até *G*, se encaminhar pacotes destinados a *G* para *A*. Da mesma forma,

Para	A	I	H	K	Novo atraso estimado de J	Interface
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	–
K	24	22	22	0	6	K
L	29	33	9	9	15	K

Figura 5.7 (a) Uma rede. (b) Entrada de *A*, *I*, *H*, *K* e a nova tabela de roteamento para *J*.

ele calcula o atraso para *G* via *I*, *H* e *K* como 41 (31 + 10), 18 (6 + 12) e 37 (31 + 6) ms, respectivamente. O melhor desses valores é 18; portanto, *J* cria uma entrada em sua tabela de roteamento indicando que o atraso até *G* é 18 ms e que a rota a ser utilizada passa por *H*. O mesmo cálculo é feito para todos os outros destinos, com a nova tabela de roteamento mostrada na última coluna da figura.

O problema da contagem ao infinito

O estabelecimento de rotas para os melhores caminhos pela rede é chamado de **convergência**. O roteamento por vetor de distância é útil como uma técnica simples para os roteadores calcularem coletivamente os caminhos mais curtos, mas tem um sério inconveniente na prática: apesar de convergir para a resposta correta, ele pode fazê-lo muito lentamente. Em particular, ele reage com rapidez às boas notícias, mas reage lentamente às más. Imagine um roteador cuja melhor rota até o destino *X* seja grande. Se, na próxima troca, o vizinho *A* repentinamente relatar um pequeno atraso até *X*, o roteador deixará de usar a interface que vai até *A* e enviará o tráfego para *X*. Em uma troca de vetores, a boa notícia sempre é processada.

Para ver a que velocidade as boas notícias se propagam, considere a rede de cinco nós (linear) da Figura 5.8, na qual a métrica para calcular o atraso é o número de hops. Suponha que *A* inicialmente esteja inativo e que todos os outros roteadores saibam disso. Em outras palavras, todos eles registraram o atraso até *A* como infinito.

Quando *A* está ativo, os outros roteadores tomam conhecimento dele por meio de trocas de vetores. Para simplificar, vamos supor que exista um gongo gigantesco em algum lugar e que ele seja tocado periodicamente para dar início a uma troca de vetores em todos os roteadores ao mesmo tempo. No momento da primeira troca, *B* toma conhecimento de que seu vizinho da esquerda tem atraso zero até *A*. Agora, *B* cria uma entrada em sua tabela de roteamento, indicando que *A* está a um hop de distância à esquerda. Todos os outros roteadores continuam imaginando que *A* está inativo. Nesse momento, as entradas da tabela de roteamento correspondentes a *A* são as que estão ilustradas na segunda linha da Figura 5.8(a). Na troca seguinte, *C* descobre que *B* tem um caminho de comprimento 1 até *A* e, portanto, atualiza sua tabela de roteamento para indicar um caminho de comprimento 2, mas *D* e *E* só detectam as boas notícias mais tarde. É claro que as boas notícias estão sendo espalhadas na velocidade de um hop por troca. Em uma rede cujo caminho mais longo tenha o comprimento de *N* hops, dentro de *N* trocas todos saberão quais enlaces e roteadores foram recentemente reativados.

Agora, vamos considerar a situação da Figura 5.8(b), em que todos os enlaces e roteadores estão inicialmente ativos. Os roteadores *B*, *C*, *D* e *E* têm distâncias até *A* iguais a 1, 2, 3 e 4, respectivamente. De repente, *A* é desativado, ou então a linha entre *A* e *B* é interrompida (o que efetivamente é o mesmo, do ponto de vista de *B*).

Na primeira troca de pacotes, *B* nada detecta vindo de *A*. Felizmente, *C* informa: "Não se preocupe. Tenho um caminho até *A* de comprimento 2". Pouco adianta *B* saber que o caminho de *C* passa pelo próprio *B*. Apesar de *B* saber disso, *C* pode ter dez enlaces de saída, todos com caminhos independentes até *A* de comprimento 2. Por conseguinte, *B* agora imagina que pode alcançar *A* via *C*, com um comprimento de caminho igual a 3. *D* e E não atualizam suas entradas correspondentes a *A* na primeira troca.

Na segunda troca, *C* percebe que cada um de seus vizinhos alega ter um caminho até *A* de comprimento 3. Ele seleciona um desses caminhos ao acaso e torna 4 sua nova distância até *A*, como mostra a terceira fileira da Figura 5.8(b). As trocas subsequentes produzem o histórico mostrado no restante da Figura 5.8(b).

A partir dessa figura, deve ficar claro por que as más notícias se propagam lentamente: nenhum roteador tem um valor maior que uma unidade a mais que o valor mínimo de todos os seus vizinhos. Gradualmente, todos os roteadores seguem seu caminho até infinito, mas o número de trocas necessárias depende do valor numérico utilizado para infinito. Por essa razão, é melhor definir infinito como o caminho mais longo mais uma unidade.

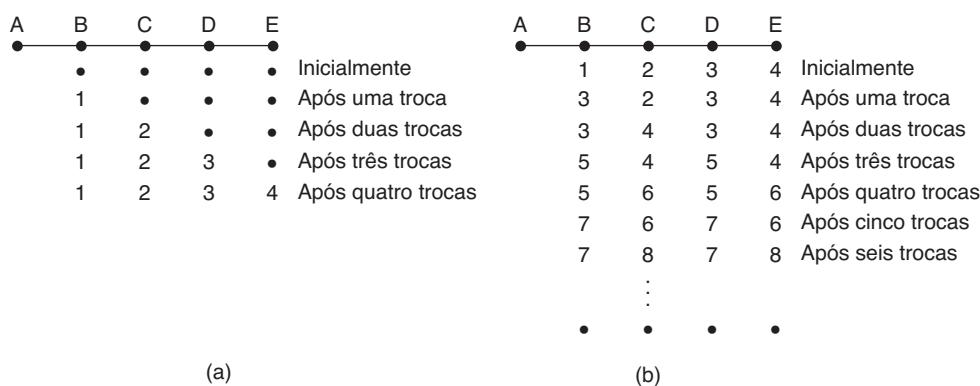



Figura 5.8 | O problema da contagem ao infinito.

Não é de surpreender totalmente que esse problema seja conhecido como problema da **contagem ao infinito**. Houve algumas tentativas de resolvê-lo, por exemplo, impedindo que os roteadores anunciassem seus melhores caminhos recém-descobertos aos seus vizinhos, em um processo conhecido como reversão envenenada, descrita na RFC 1058. Contudo, nenhuma dessas heurísticas funcionou bem na prática, apesar dos nomes bonitos. O núcleo do problema é que, quando *X* informa a *Y* que tem um caminho em algum lugar, *Y* não tem como saber se ele próprio está no caminho.

5.2.5 Roteamento de estado de enlace

O roteamento por vetor de distância foi utilizado na ARPANET até 1979, quando foi substituído pelo roteamento de estado de enlace. O problema principal que causou sua retirada foi que o algoritmo geralmente levava muito tempo para convergir (em decorrência do problema da contagem ao infinito). Por conseguinte, ele foi substituído por um algoritmo inteiramente novo, agora chamado **roteamento de estado de enlace**. Variantes do roteamento de estado de enlace, chamadas IS-IS e OSPF, são os algoritmos de roteamento mais usados dentro de grandes redes e da Internet atualmente.

A ideia por trás do roteamento de estado de enlace é simples e pode ser estabelecida em cinco partes. Cada roteador deve fazer o seguinte:

1. Descobrir seus vizinhos e aprender seus endereços de rede.
2. Medir a distância ou o custo até cada um de seus vizinhos.
3. Criar um pacote que informe tudo o que ele acabou de aprender.
4. Enviar esse pacote e receber pacotes de todos os outros roteadores.
5. Calcular o caminho mais curto até cada um dos outros roteadores.

Com efeito, a topologia completa é distribuída para todos os outros roteadores. Em seguida, o algoritmo de Dijkstra pode ser usado em cada roteador para encontrar o caminho mais curto até cada um dos outros roteadores. A seguir, estudaremos cada uma dessas cinco etapas em detalhes.

Conhecendo os vizinhos

Quando um roteador é iniciado, sua primeira tarefa é aprender quem são seus vizinhos. Esse objetivo é alcançado enviando-se um pacote HELLO especial em cada linha ponto a ponto. O roteador da outra extremidade deve enviar de volta uma resposta, informando quem é. Esses nomes devem ser globalmente exclusivos, pois, quando mais tarde um roteador distante ouvir que esses três roteadores estão todos conectados a *F*, é essencial que ele possa determinar se os três representam o mesmo *F*.

Quando dois ou mais roteadores estão conectados por um enlace de broadcast (por exemplo, em um switch, em um anel ou na Ethernet clássica), a situação é um pouco mais complicada. A Figura 5.9(a) ilustra uma LAN broadcast à qual três roteadores, *A*, *C* e *F*, estão diretamente conectados. Cada um desses roteadores está conectado a um ou mais roteadores adicionais, como mostra a figura.

A LAN broadcast oferece conectividade entre cada par de roteadores conectados. Porém, a modelagem da LAN com muitos enlaces ponto a ponto aumenta o tamanho da topologia e leva ao desperdício de mensagens. Uma forma melhor de modelar a LAN é considerá-la um nó, conforme mostra a Figura 5.9(b). Aqui, introduzimos um novo nó artificial, *N*, ao qual *A*, *C* e *F* estão conectados. Um **roteador** na LAN é selecionado para desempenhar o papel de *N* no protocolo de roteamento. A possibilidade de ir de *A* até *C* na LAN é representada aqui pelo caminho *ANC*.

Como medir o custo do enlace

O algoritmo de roteamento de estado de enlace exige que cada enlace tenha uma medida da distância ou de custo para encontrar os caminhos mais curtos. O custo para alcançar os vizinhos pode ser definido automaticamente,

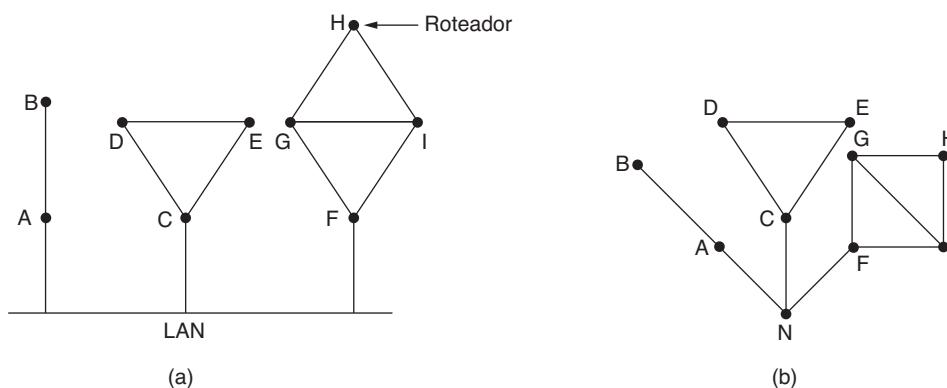



Figura 5.10 | (a) Nove roteadores e uma LAN broadcast. (b) Um modelo de grafo de (a).

ou então configurado pelo operador da rede. Uma escolha comum é tornar o custo inversamente proporcional à largura de banda do enlace. Por exemplo, a Ethernet de 1 Gbps pode ter um custo de um e a Ethernet de 100 Mbps, um custo de dez. Isso faz com que os caminhos de maior capacidade se tornem as melhores escolhas.

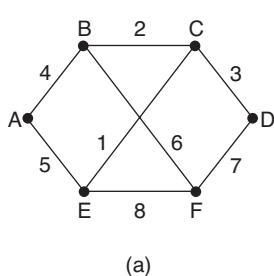
Se a rede estiver espalhada geograficamente, o atraso dos enlaces pode ser computado no custo, de modo que os caminhos por enlaces mais curtos são as melhores escolhas. A forma mais simples de determinar esse atraso é enviar um pacote especial ECHO/REPLY pelo enlace, que o outro lado devolve imediatamente. Medindo o tempo de ida e volta e dividindo-o por dois, o roteador transmissor pode obter uma estimativa razoável do atraso.

Como criar pacotes de estado de enlace

Uma vez obtidas as informações necessárias para a troca, a próxima etapa é cada roteador criar um pacote que contenha todos os dados. O pacote começa com a identidade do transmissor, seguida por um número de sequência e pelo tempo de vida (TTL, a ser descrito mais adiante) e por uma lista de vizinhos. É fornecido o custo até cada vizinho. Um exemplo de rede é apresentado na Figura 5.10(a), sendo os custos mostrados como rótulos nas linhas. Os pacotes de estado de enlace correspondentes a todos os seis roteadores são mostrados na Figura 5.10(b).

É fácil criar os pacotes de estado de enlace. Difícil é determinar quando criá-los. Uma possibilidade é criá-los periodicamente, ou seja, em intervalos regulares. Outra possibilidade é criá-los durante a ocorrência de algum evento significativo, como uma interface ou vizinho que sai do ar, entra em atividade novamente ou altera suas propriedades de forma considerável.

Distribuição dos pacotes de estado de enlace

A parte mais complicada do algoritmo é distribuir os pacotes de estado de enlace. Todos os roteadores precisam obter todos os pacotes de estado de enlace de modo rápido e confiável. Se diferentes roteadores estiverem usando diferentes versões da topologia, as rotas que eles calculam podem ter inconsistências, como loops, máquinas inacessíveis e outros problemas.

Primeiro, descreveremos o algoritmo básico de distribuição. Depois, vamos aperfeiçoá-lo. A ideia fundamental é usar o flooding para distribuir os pacotes de estado de enlace para todos os roteadores. Para manter o controle do flooding, cada pacote contém um número de sequência que é incrementado para cada novo pacote enviado. Os roteadores controlam todos os pares (roteador de origem, sequência) que veem. Quando é recebido, o novo pacote de estado de enlace é conferido na lista de pacotes já verificados. Se for novo, será encaminhado a todas as interfaces, exceto àquela por onde chegou. Se for uma cópia, o pacote será descartado. Se um pacote recebido tiver número de sequência mais baixo que o mais alto número de sequência detectado até o momento, ele será rejeitado e considerado obsoleto, pois o roteador terá dados mais recentes.

Esse algoritmo apresenta alguns problemas, mas eles são contornáveis. Primeiro, se os números de sequência se repetirem, haverá confusão. A solução aqui é usar um número de sequência de 32 bits. Com um pacote de estado de enlace por segundo, seriam necessários 137 anos para um número se repetir; portanto, essa possibilidade pode ser ignorada.

Segundo, se um roteador apresentar falha, ele perderá o controle de seu número de sequência. Se ele começar de novo em zero, o pacote seguinte será rejeitado por ser considerado uma cópia.

Terceiro, se um número de sequência for adulterado e o número 65.540 for recebido no lugar do número 4 (um erro de 1 bit), os pacotes de 5 a 65.540 serão rejeitados como obsoletos, pois 65.540 será considerado o número de sequência atual.

A solução para todos esses problemas é incluir o tempo de vida ou TTL de cada pacote após o número de sequência e decrementá-lo uma vez por segundo. Quando o TTL atingir zero, as informações desse roteador serão descartadas. Normalmente, um novo pacote chega, digamos, a cada 10 segundos; logo, as informações do roteador só alcançarão o timeout (tempo-limite) quando um roteador estiver inativo (ou quando seis pacotes consecutivos se perderem, um evento improvável). O campo *TTL* também é decrementado por cada roteador durante o processo inicial de flooding, para garantir que nenhum pacote será perdido e durará

Figura 5.10 | (a) Uma rede. (b) Os pacotes de estado de enlace correspondentes a essa rede.

um período indefinido (um pacote cujo TTL for zero será descartado).

Alguns aprimoramentos nesse algoritmo o tornam mais resistente. Quando um pacote de estado de enlace chega a um roteador para o flooding, ele não é imediatamente enfileirado para transmissão. Em vez disso, ele é colocado em uma área de retenção para aguardar um pouco, caso mais enlaces estejam chegando ou saindo. Se outro pacote de estado de enlace da mesma origem chegar antes da transmissão do primeiro pacote, seus números de sequência serão comparados. Se forem iguais, a cópia será descartada. Se forem diferentes, o mais antigo será descartado. Para evitar erros nos enlaces, todos os pacotes de estado de enlace são confirmados.

A estrutura de dados utilizada pelo roteador *B* da rede mostrada na Figura 5.10(a) é representada na Figura 5.11. Cada interface aqui corresponde a um pacote de estado de enlace recém-chegado, mas ainda não totalmente processado. A tabela registra a origem do pacote, seu número de sequência e TTL, além dos dados correspondentes. Além disso, há flags de transmissão e confirmação para cada uma das três interfaces de *B* (para *A*, *C* e *F*, respectivamente). Flags de transmissão significam que o pacote deve ser enviado no enlace indicado. Flags de confirmação significam que ele deve ser confirmado ali.

Na Figura 5.11, o pacote de estado de enlace de *A* chega diretamente; portanto, ele deve ser enviado para *C* e *F* e confirmado por *A*, conforme indicam os bits das flags. Da mesma forma, o pacote proveniente de *F* deve ser encaminhado para *A* e *C*, e confirmado por *F*.

Entretanto, a situação com o terceiro pacote, proveniente de *E*, é diferente. Ele chegou duas vezes, uma vez por *EAB* e outra por *EFB*. Consequentemente, ele só precisa ser enviado para *C*, mas deve ser confirmado por *A* e *F*, conforme indicam os bits.

Se uma cópia for recebida enquanto o original ainda estiver no buffer, os bits deverão ser alterados. Por exemplo, se uma cópia do estado de *C* chegar de *F* antes de a quarta entrada da tabela ter sido encaminhada, os seis bits serão alterados para 100011, indicando que o pacote deve ser confirmado para *F*, mas não deve ser enviado para lá.

Como calcular as novas rotas

Uma vez que um roteador tenha acumulado um conjunto completo de pacotes de estado de enlace, ele poderá criar o grafo da rede inteira, pois todos os enlaces estarão representados. Na verdade, todo enlace será representado duas vezes, uma vez em cada sentido. Os diferentes sentidos podem até mesmo ter diferentes custos. Então, os cálculos de caminho mais curto poderão achar diferentes caminhos do roteador *A* para *B* e vice-versa.

Agora o algoritmo de Dijkstra pode ser executado localmente com a finalidade de criar os caminhos mais curtos até todos os destinos possíveis. Os resultados desse algoritmo dizem ao roteador qual enlace utilizar para alcançar cada destino. Essa informação é inserida nas tabelas de roteamento, e a operação normal pode ser retomada.

Em comparação com o roteamento por vetor de distância, o roteamento de estado de enlace requer mais memória e cálculos. Para uma rede com *n* roteadores, cada qual com *k* vizinhos, a memória necessária para armazenar os dados de entrada é proporcional a *kn*, que, no mínimo, é tão grande quanto a listagem da tabela de roteamento que relaciona todos os destinos. Além disso, o tempo de cálculo cresce mais rápido que *kn*, mesmo com estruturas de dados mais eficientes, um problema nas grandes redes. Apesar disso, em muitas situações práticas, o roteamento de estado de enlace funciona muito bem, pois não sofre com os problemas de convergência lenta.

O roteamento de estado de enlace é muito utilizado em redes reais; portanto, vale a pena fazer alguns comentários sobre exemplos de protocolos que o utilizam. Muitos ISPs utilizam o protocolo de **estado de enlace intersistemas**, ou **IS-IS** (**Intermediate System-Intermediate System**) (Oran, 1990). Ele foi projetado para uma antiga rede chamada DECnet, adotado mais tarde pela ISO para uso com os protocolos OSI e depois modificado para lidar com outros protocolos também, entre os quais se destaca o IP. O **OSPF** (**Open Shortest Path First**) é o outro principal protocolo de estado de enlace. Ele foi projetado pelo IETF muitos anos depois do IS-IS e adotou muitas das inovações projetadas para aquele. Entre essas inovações estão as seguintes: um método de

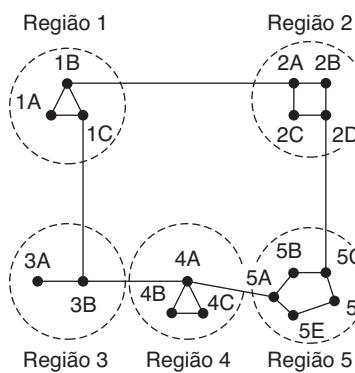
			Flags de envio			Flags de ACK			
Origem	Seq.	TTL	A	C	F	A	C	F	Dados
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

Figura 5.11 | O buffer de pacotes correspondente ao roteador *B* da Figura 5.10(a).

autoestabilização de atualizações de estado de enlace por flooding, o conceito de roteador designado em uma LAN e o método de cálculo e suporte ao split horizon, além de várias métricas. Consequentemente, há pouca diferença entre o IS-IS e o OSPF. A mais importante delas é que o IS-IS pode transportar simultaneamente informações sobre vários protocolos da camada de rede (por exemplo, IP, IPX e AppleTalk), um recurso que o OSPF não apresenta. Essa vantagem é especialmente valiosa em grandes ambientes de vários protocolos. Voltaremos a ver o OSPF na Seção 5.6.6.

Um comentário geral sobre algoritmos de roteamento também é pertinente aqui. Estado de enlace, vetor de distância e outros algoritmos contam com o processamento em todos os roteadores para calcular as rotas. Problemas com o hardware ou com o software, até mesmo com um pequeno número de roteadores, podem causar grandes complicações na rede. Por exemplo, se um roteador alegar ter um enlace que na realidade não tem, ou se esquecer de um enlace que tem, o grafo da rede ficará incorreto. Se um roteador deixar de encaminhar pacotes ou danificá-los enquanto os encaminhar, a rota não funcionará como se espera. Por fim, se a memória do roteador se esgotar ou se ele calcular o roteamento incorretamente, as falhas serão inúmeras. À medida que a rede cresce até a faixa de dezenas ou centenas de milhares de nós, a probabilidade de algum roteador falhar ocasionalmente deixará de ser desprezível. O truque é tentar limitar os danos quando acontecer o inevitável. Perlman (1988) discute em detalhes esses problemas e suas possíveis soluções.

5.2.6 Roteamento hierárquico

À medida que as redes aumentam de tamanho, as tabelas de roteamento dos roteadores crescem proporcionalmente. Não apenas a memória do roteador é consumida por tabelas cada vez maiores, mas também é necessário dedicar maior tempo de CPU para percorrê-las e mais largura de banda para enviar relatórios de status sobre elas. Em determinado momento, a rede pode crescer até o ponto em que deixará de ser viável cada roteador ter uma entrada correspondente a cada outro roteador, de forma que o roteamento terá de ser feito de forma hierárquica, como na rede telefônica.

Quando o roteamento hierárquico for utilizado, os roteadores serão divididos naquilo que denominaremos **regiões**, com cada roteador conhecendo todos os detalhes sobre como rotear pacotes para destinos dentro de sua própria região, mas sem conhecer nada sobre a estrutura interna de outras regiões. Quando diferentes redes estão interconectadas, é natural que cada uma seja vista como uma região separada, a fim de liberar os roteadores de uma rede da necessidade de conhecer a estrutura topológica das outras redes.

No caso de redes muito grandes, uma hierarquia de dois níveis talvez seja insuficiente; provavelmente será necessário reunir as regiões em agrupamentos (clusters), os agrupamentos em zonas, as zonas em grupos etc., até faltarem nomes para os agregados. Como exemplo de uma hierarquia de vários níveis, vejamos como um pacote poderia ser roteado de Berkeley, na Califórnia, até Malindi, no Quênia. O roteador de Berkeley conheceria a topologia detalhada da Califórnia, mas enviaria todo o tráfego de fora do estado para o roteador de Los Angeles. Este seria capaz de rotear o tráfego para outros roteadores domésticos, mas enviaria todo o tráfego destinado a outros países para Nova York. O roteador de Nova York seria programado de modo a direcionar todo o tráfego para o roteador no país de destino responsável pelo tratamento do tráfego vindo do exterior, digamos, em Nairóbi. Por fim, o pacote seguiria seu caminho descendente pela árvore no Quênia até chegar a Malindi.

A Figura 5.12 fornece um exemplo quantitativo do roteamento em uma hierarquia de dois níveis com cinco regiões. A tabela de roteamento completa do roteador *1A* tem 17 entradas, como mostra a Figura 5.12(b). Quando o roteamento for feito hierarquicamente, como na Figura 5.12(c), haverá entradas para todos os roteadores locais, como antes, mas todas as outras regiões terão sido condensadas em um único roteador; portanto, todo o tráfego destinado à região 2 passará pela interface *1B-2A*, mas o restante do tráfego remoto utilizará a interface *1C-3B*. O roteamento hierárquico reduz a tabela de 17 para 7 entradas. À medida que cresce a relação entre o número de regiões e o número de roteadores por região, a economia de espaço na tabela aumenta.

Infelizmente, esses ganhos em espaço não são gratuitos. Há um preço a ser pago: um aumento no comprimento do caminho. Por exemplo, a melhor rota de *1A* até *5C* passa pela região 2; no entanto, com o roteamento hierárquico, todo o tráfego destinado à região 5 segue pela região 3, porque essa é a melhor opção para a maior parte dos destinos na região 5.

Quando uma única rede se torna muito extensa, surge uma questão interessante: "Quantos níveis a hierarquia deve ter?". Por exemplo, considere uma rede com 720 roteadores. Se não houver hierarquia, cada roteador precisará de 720 entradas na tabela de roteamento. Se a rede for particionada em 24 regiões de 30 roteadores cada uma, cada roteador precisará de 30 entradas locais e mais 23 entradas remotas, perfazendo um total de 53 entradas. Se for escolhida uma hierarquia de três níveis com oito agrupamentos, cada um deles contendo nove regiões de dez roteadores, cada roteador precisará de dez entradas para roteadores locais, oito entradas para roteamento até outras regiões dentro de seu próprio agrupamento e sete entradas para agrupamentos distantes, perfazendo um total de 25 entradas. Kamoun e Kleinrock (1979) descobriram que o

(a)

Tabela completa para 1A

Dest.	Interface	Hops
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

(b)

Tabela hierárquica para 1A

Dest.	Interface	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

(c)

Figura 5.12 | Roteamento hierárquico.

número ideal de níveis para uma rede com *N* roteadores é ln *N*, exigindo um total de *e* ln *N* entradas por roteador. Eles também demonstraram que o aumento na extensão do caminho médio efetivo causado pelo roteamento hierárquico é suficientemente pequeno, de forma que, de modo geral, é aceitável.

5.2.7 Roteamento por broadcast

Em algumas aplicações, os hosts precisam enviar mensagens a muitos ou a todos os outros hosts. Por exemplo, um serviço de distribuição de relatórios sobre o tempo, atualizações do mercado de ações ou programas de rádio ao vivo poderiam funcionar melhor por meio do envio das informações a todas as máquinas, permitindo que as que estivessem interessadas lessem os dados. O envio de um pacote a todos os destinos simultaneamente é chamado **broadcasting** (difusão). Foram propostos vários métodos para implementar esse recurso.

Um método de broadcasting que não exige recursos especiais da rede permite à origem simplesmente enviar um pacote específico a cada destino. O método não só desperdiça largura de banda mas também exige que a origem tenha uma lista completa de todos os destinos. Esse método não é desejável na prática, embora se aplique de forma generalizada.

Uma melhora é o **roteamento para vários destinos**. Se esse método for utilizado, cada pacote conterá uma lista de destinos ou um mapa de bits indicando os destinos desejados. Quando um pacote chega a um roteador, este verifica todos os destinos para determinar o conjunto de interfaces de saída que serão necessárias. (Uma interface de saída será necessária se for a melhor rota a pelo menos um dos destinos.) O roteador gera uma nova cópia do pacote para cada interface de saída a ser utilizada e inclui em cada pacote somente os destinos que usarão a interface. Na verdade, o conjunto de destinos é particionado entre as interfaces de saída. Após um número suficiente de hops, cada pacote será transportado somente para um destino e poderá ser tratado como um pacote normal. O roteamento para vários destinos é como utilizar pacotes endereçados separadamente, exceto pelo fato de que, quando vários pacotes tiverem de seguir a mesma rota, um deles pagará toda a passagem e os restantes viajarão de graça. A largura de banda da rede, portanto, é usada de modo mais eficiente. Contudo, esse esquema ainda requer que a origem conheça todos os destinos, e ainda é muito trabalhoso para um roteador determinar para onde enviar um pacote para vários destinos, da mesma forma que para múltiplos pacotes distintos.

Já vimos uma técnica de roteamento por broadcast melhor: o flooding. Quando implementado com um número de sequência por origem, o flooding utiliza enlaces de modo eficiente com uma regra de decisão nos roteadores que é relativamente simples. Embora o flooding não seja muito adequado para a comunicação ponto a ponto comum, ele deve ser levado em consideração no broadcasting. Porém, podemos fazer ainda melhor quando as rotas de caminho mais curto para os pacotes normais tiverem sido calculadas.

A ideia do **encaminhamento pelo caminho inverso** é elegante e muito simples, uma vez que é compreendi-

da (Dalal e Metcalfe, 1978). Quando um pacote de broadcast chega a um roteador, este verifica se o pacote chegou pela interface que originalmente utilizou para o envio de pacotes de broadcast *para* um destino. Em caso afirmativo, há uma excelente possibilidade de que o pacote de broadcast tenha seguido a melhor rota a partir do roteador e seja, portanto, a primeira cópia a chegar a ele. Se for esse o caso, o roteador encaminhará cópias do pacote para todas as interfaces, exceto para aquela por onde ele chegou. Porém, se o pacote de broadcast chegou por um enlace diferente do preferido para alcançar o destino, o pacote é descartado como uma provável cópia.

Um exemplo do algoritmo de encaminhamento pelo caminho inverso é mostrado na Figura 5.13. A parte (a) mostra uma rede, a parte (b) mostra uma árvore de escoamento para o roteador *I* dessa rede e a parte (c) mostra como funciona o algoritmo de encaminhamento pelo caminho inverso. No primeiro hop, *I* envia pacotes para *F*, *H*, *J* e *N*, como indica a segunda interface da árvore. Cada um desses pacotes chega ao caminho preferencial para *I* (supondo que o caminho preferencial acompanhe a árvore de escoamento) e é, então, indicado por um círculo em torno da letra. No segundo hop, são gerados oito pacotes, dois por cada um dos roteadores que receberam um pacote no primeiro hop. Por sua vez, todos os oito pacotes chegam a roteadores não visitados anteriormente, e cinco deles chegam ao longo da interface preferencial. Dos seis pacotes gerados no terceiro hop, somente três chegam pelo caminho preferencial (em *C*, *E* e *K*); os outros são cópias. Depois de cinco hops e 24 pacotes, o broadcasting termina, em comparação com quatro hops e 14 pacotes que haveria se a árvore de escoamento fosse seguida exatamente.

A principal vantagem do encaminhamento pelo caminho inverso é que ele, ao mesmo tempo, é eficiente e fácil de implementar. Ele envia o pacote de broadcast por cada enlace apenas uma vez em cada sentido, assim como no flooding, enquanto exige apenas que os roteadores saibam como alcançar todos os destinos, sem ter de lembrar dos números de sequência (ou usar outros mecanismos para interromper o flooding) ou listar todos os destinos no pacote.

Nosso último algoritmo de broadcast melhora o comportamento do encaminhamento pelo caminho inverso. Ele faz uso explícito da árvore de escoamento — ou qualquer outra árvore spanning tree — para o roteador que inicia o broadcast. Uma spanning tree é uma **árvore** que é um subconjunto da rede que inclui todos os roteadores, mas não contém loops. As árvores de escoamento são do tipo spanning tree. Se cada roteador souber quais de suas interfaces pertencem à spanning tree, ele poderá copiar um pacote de broadcast da entrada para todas as interfaces da spanning tree, exceto para aquela pela qual o pacote chegou. Esse método faz excelente uso da largura de banda, gerando o número mínimo absoluto de pacotes necessários para realizar essa tarefa. Na Figura 5.13, por exemplo, quando a árvore de escoamento da parte (b) é usada como spanning tree, o pacote de broadcast é enviado com o mínimo de 14 pacotes. O único problema é que cada roteador deve ter conhecimento de uma spanning tree para que o método seja aplicável. Às vezes essas informações estão disponíveis (por exemplo, com o roteamento de estado de enlace, em que todos os roteadores conhecem a topologia completa e, por isso, podem calcular uma spanning tree), mas às vezes não (por exemplo, no caso do roteamento por vetor de distância).

5.2.8 Roteamento por multicast

Algumas aplicações, como jogos com mais de um participante ou vídeo ao vivo de um evento esportivo, transmitido para muitos locais de exibição, enviam pacotes para vários receptores. A menos que o grupo seja muito pequeno, o envio de um pacote distinto a cada receptor é dispendioso. Por outro lado, o broadcasting de um pacote é um desperdício se o grupo tiver, digamos, mil máquinas em uma rede de um milhão de nós, de modo que a maioria dos receptores não está interessada na mensagem (ou, pior ainda, os receptores estão definitivamente interessados, mas não veem a mensagem). Desse modo, precisamos de um meio para enviar mensagens a grupos bem definidos que têm um tamanho numericamente grande, mas que são pequenos em comparação à rede como um todo.

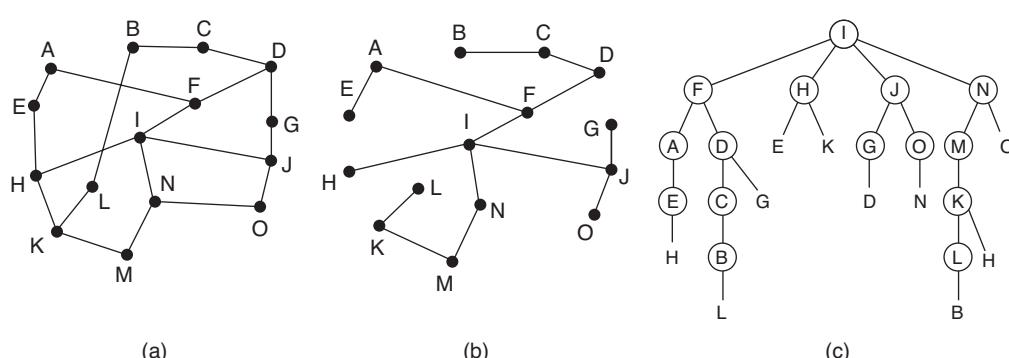



Figura 5.13 | Encaminhamento pelo caminho inverso. (a) Uma rede. (b) Uma árvore de escoamento. (c) A árvore construída por encaminhamento pelo caminho inverso.

O envio de uma mensagem a um desses grupos denomina-se **multicasting** (multidifusão) e seu algoritmo de roteamento é chamado **roteamento por multicasting**. O multicasting exige o gerenciamento de grupos. Será preciso usar algum método para criar e destruir grupos, e para identificar quais roteadores são membros de um grupo. O modo como essas tarefas serão realizadas não interessa ao algoritmo de roteamento. Por enquanto, vamos considerar que cada grupo é identificado por um endereço de multicast e que os roteadores conhecem os grupos aos quais eles pertencem. Retornaremos ao assunto de membros de grupo quando descrevermos a camada de rede da Internet, na Seção 5.6.

Os esquemas de roteamento por multicast se baseiam nos esquemas de roteamento por broadcast que já estudamos, enviando pacotes pelas spanning trees para entregá-los aos membros do grupo enquanto utilizam a largura de banda com eficiência. Porém, a melhor spanning tree a ser usada depende se o grupo é denso, com receptores espalhados pela maior parte da rede, ou esparso, com boa parte da rede não pertencente ao grupo. Nesta seção, vamos considerar esses dois casos.

Se o grupo é denso, o broadcast é um bom ponto de partida, pois ele leva o pacote com eficiência a todas as partes da rede. Porém, o broadcast alcançará alguns roteadores que não fazem parte do grupo, o que é um desperdício. A solução explorada por Deering e Cheriton (1990) é podar a spanning tree por broadcast, removendo os enlaces que não levam a membros. O resultado é uma spanning tree de multicast eficiente.

Como um exemplo, considere os dois grupos, 1 e 2, na rede mostrada na Figura 5.14(a). Alguns roteadores estão associados a hosts que pertencem a um ou a ambos os grupos, como indica a figura. Uma spanning tree correspondente ao roteador situado mais à esquerda é mostrada na Figura 5.14(b). Essa árvore pode ser usada para o broadcast, mas é um exagero para o multicast, como podemos ver pelas duas versões podadas que aparecem em seguida. Na Figura 5.14(c), todos os enlaces que não levam a hosts que são membros do grupo 1 foram removidos. O resultado é a spanning tree por multicast para o roteador mais à esquerda enviado para o grupo 1. Os pacotes são encaminhados apenas ao longo dessa spanning tree, o que é mais eficiente que a árvore de broadcast, pois existem sete enlaces em vez de dez. A Figura 5.14(d) mostra a spanning tree de multicast após a poda para o grupo 2. Ela também é eficiente, com apenas cinco enlaces dessa vez. Percebe-se, assim, que diferentes grupos de multicast têm diferentes spanning trees.

Existem vários métodos que podem ser usados para podar uma spanning tree. O mais simples pode ser usado se o roteamento de estado de enlace for empregado e cada roteador estiver ciente da topologia completa, inclusive quais hosts pertencem a cada um dos grupos. Cada roteador pode, então, construir sua própria spanning tree podada para cada transmissor para o grupo em questão, normalmente construindo uma árvore de escoamento para o transmissor e, depois, removendo todos os enlaces que não conectam membros do grupo ao nó de escoamento. O **MOSPF** (**Multicast**

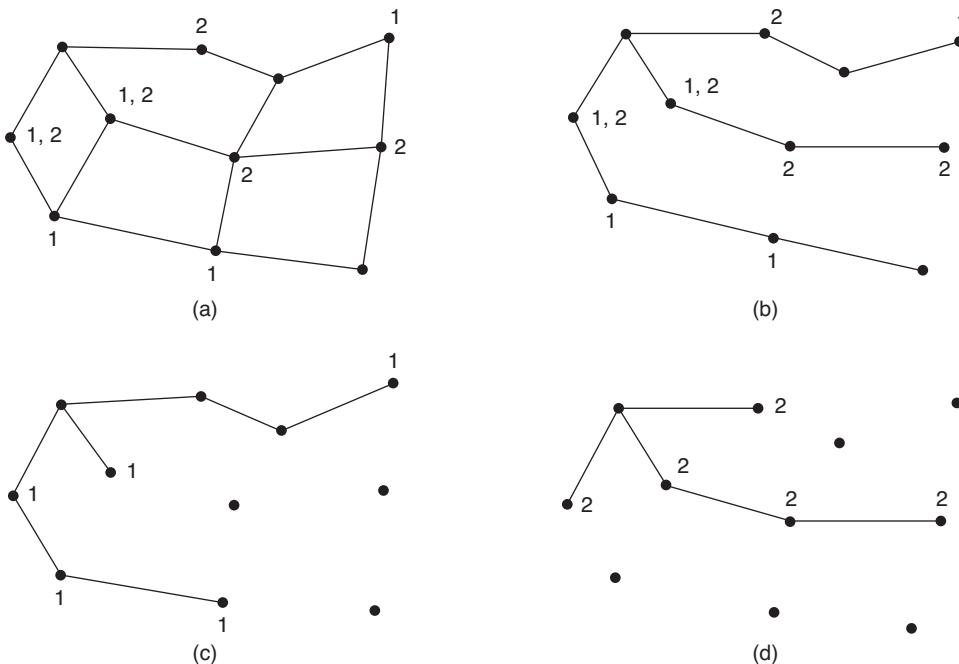



Figura 5.14 | (a) Uma rede. (b) Uma spanning tree para o roteador da esquerda. (c) Uma árvore de multicast para o grupo 1. (d) Uma árvore de multicast para o grupo 2.

OSPF) é um exemplo de um protocolo de estado de enlace que funciona dessa maneira (Moy, 1994).

Quando se emprega o roteamento por vetor de distância, é possível utilizar outra estratégia de poda. O algoritmo básico é o encaminhamento pela nota inversa. Entretanto, sempre que um roteador sem hosts pertencentes a um grupo específico e sem conexões com outros roteadores recebe uma mensagem de multicast relacionada a esse grupo, ele responde com uma mensagem PRUNE, informando ao transmissor que este não deve enviar mais mensagens de multicast para esse grupo. Quando um roteador sem membros de grupos entre seus próprios hosts recebe tais mensagens em todas as suas interfaces para as quais ele envia o multicast, ele também pode responder com uma mensagem PRUNE. Assim, a rede será podada recursivamente. O protocolo de roteamento multicast por vetor de distância, ou **DVMRP** (**Distance Vector Multicast Routing Protocol**), é um exemplo de protocolo de roteamento de multicast que funciona dessa maneira (Waitzman et al., 1988).

A poda resulta em spanning trees eficientes, que usam apenas os enlaces realmente necessários para alcançar os membros do grupo. Uma desvantagem potencial desse algoritmo é que ele gera muito trabalho para os roteadores, especialmente em redes maiores. Suponha que uma rede tenha n grupos, cada qual com uma média de m nós. Em cada roteador e para cada grupo, devem ser armazenadas m spanning trees podadas, perfazendo um total de mn árvores. Por exemplo, a Figura 5.14(c) mostra a spanning tree para o roteador mais à esquerda enviando para o grupo 1. A spanning tree para o roteador mais à direita enviando para o grupo 1 (que não aparece) será muito diferente, pois os pacotes retornarão diretamente para os membros do grupo, em vez de pelo lado esquerdo do grafo. Isso, por sua vez, significa que os roteadores devem encaminhar pacotes destinados ao grupo 1 em diferentes direções, dependendo de qual nó está enviando ao grupo. Quando existem muitos grupos grandes, com muitos transmissores, é necessário um armazenamento considerável para todas as árvores.

Um projeto alternativo utiliza **árvores baseadas em núcleo** (Ballardie et al., 1993). Aqui é calculada uma única spanning tree por grupo. Todos os roteadores concordam com uma raiz (o **núcleo**) e montam a árvore enviando um pacote de cada membro para essa raiz. A árvore é a união dos caminhos traçados por esses pacotes. A Figura 5.15(a) mostra uma árvore baseada em núcleo para o grupo 1. Para enviar uma mensagem para esse grupo, um transmissor manda um pacote ao núcleo. Quando o pacote alcança o núcleo, ele é encaminhado pela árvore. Isso pode ser visto na Figura 5.15(b) para o transmissor no lado direito da rede. Como uma otimização de desempenho, os pacotes destinados ao grupo não precisam alcançar o núcleo antes de ser transmitidos por multicast. Assim que o pacote alcança a árvore, ele pode ser encaminhado para cima, em direção à raiz, ou para baixo, para todos os outros galhos. Isso acontece para o transmissor no topo da Figura 5.15(b).

Ter uma árvore compartilhada não é ideal para todas as origens. Por exemplo, na Figura 5.15(b), o pacote do transmissor no lado direito alcança o membro do grupo superior direito por meio do núcleo em três hops, e não diretamente. A ineficiência depende do local onde o núcleo e os transmissores estão localizados, mas normalmente é razoável quando o núcleo está no meio dos transmissores. Quando existe apenas um único transmissor, como em um vídeo transmitido para um grupo, o uso do transmissor como núcleo é o ideal.

Observe também que as árvores compartilhadas podem ser uma economia importante em custos de armazenamento, envio de mensagens e computação. Cada roteador precisa manter apenas uma árvore por grupo, em vez de m árvores. Além disso, os roteadores que não fazem parte da árvore não realizam trabalho para dar suporte ao grupo. Por esse motivo, abordagens de árvore compartilhada, como as árvores baseadas em núcleo, são usadas em multicasting para grupos esparsos na Internet como parte dos protocolos populares como o **multicast independente de protocolo**, ou **PIM** (**Protocol Independent Multicast**) (Fenner et al., 2006).

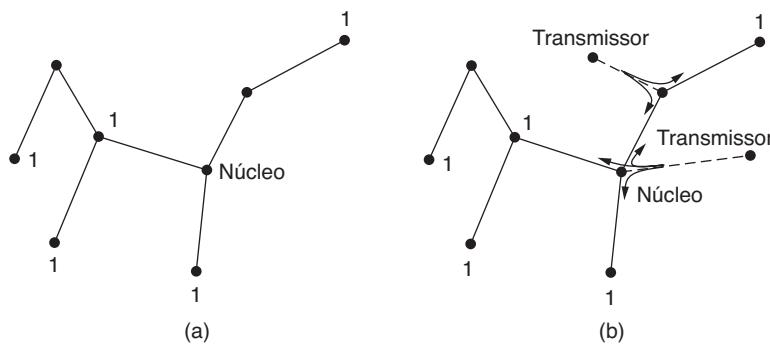



Figura 5.15 | (a) Árvore baseada em núcleo para o grupo 1. (b) Enviando para o grupo 1.

5.2.9 Roteamento por anycast

Até aqui, estudamos os modelos de entrega em que uma origem envia para um único destino (chamado **unicast**), para todos os destinos (chamado **broadcast**) e para um grupo de destinos (chamado **multicast**). Outro modelo de entrega, chamado **anycast**, às vezes também é útil. No anycast, um pacote é entregue ao membro mais próximo de um grupo (Partridge et al., 1993). Os esquemas que encontram esses caminhos são chamados **roteamento por anycast**.

Por que desejaríamos usar o anycast? Às vezes, os nós oferecem um serviço, como a hora do dia ou a distribuição de conteúdo, para o qual obter a informação correta é tudo o que importa, e não o nó que é contatado; qualquer nó servirá. Por exemplo, o anycast é usado na Internet como parte do DNS, conforme veremos no Capítulo 7.

Felizmente, não teremos de criar novos esquemas de roteamento para o anycast, pois o roteamento normal por vetor de distância e de estado de enlace pode produzir rotas de anycast. Suponha que queiramos realizar o anycast para os membros do grupo 1. Todos eles receberão no endereço "1", em vez dos diferentes endereços. O roteamento por vetor de distância distribuirá vetores normalmente, e os nós escolherão o caminho mais curto até o destino 1. Isso resultará em nós enviando para a ocorrência mais próxima do destino 1. As rotas aparecem na Figura 5.16(a). Esse procedimento funciona porque o protocolo de roteamento não observa que existem várias ocorrências do destino 1. Ou seja, ele acredita que todas as ocorrências do nó 1 são o mesmo nó, como na topologia mostrada na Figura 5.16(b).

Esse procedimento também funciona para o roteamento de estado de enlace, embora exista uma consideração adicional de que o protocolo de roteamento não deva encontrar caminhos aparentemente curtos que passam pelo nó 1. Isso resultaria em saltos pelo hiperespaço, pois as ocorrências do nó 1 são, na realidade, nós localizados em diferentes partes da rede. Contudo, os protocolos de estado de enlace já fazem essa distinção entre roteadores e hosts. Passamos por cima desse fato anteriormente porque ele não foi necessário para nossa discussão.

5.2.10 Roteamento para dispositivos móveis

Milhões de pessoas utilizam computadores enquanto viajam, desde situações verdadeiramente móveis, com dispositivos sem fio em carros em movimento, até situações nômades, em que notebooks são usados em diversos locais. Usaremos o termo **hosts móveis** para indicar essa categoria, ao contrário dos hosts fixos, que nunca se movem. Cada vez mais as pessoas desejam permanecer conectadas onde quer que estejam no mundo, como se estivessem em sua casa. Esses hosts móveis criam uma nova complicação: antes de rotear um pacote para um host móvel, primeiro a rede precisa localizá-lo.

O modelo do mundo que consideraremos é aquele em que todos os hosts têm um **local fixo** permanente, que nunca muda. Os hosts também têm um endereço local permanente que pode ser usado para determinar seus locais fixos, de modo semelhante à forma como o número de telefone 1-212-5551212 indica que se trata dos Estados Unidos (código de país 1) e de Manhattan (212). O objetivo do roteamento em sistemas com hosts móveis é tornar possível o envio de pacotes a hosts móveis que estejam usando seus endereços locais e fazer os pacotes alcançar esses hosts de forma eficiente, onde quer que eles possam estar. Evidentemente, o problema é localizá-los.

Precisamos discutir um pouco sobre esse modelo. Um modelo diferente seria recalcular as rotas à medida que o host móvel se move e a topologia muda. Poderíamos, então, simplesmente usar os esquemas de roteamento descritos anteriormente nesta seção. Porém, com um número cada vez maior de hosts móveis, esse modelo logo levaria a uma rede inteira calculando incessantemente as novas rotas. O uso de endereços locais reduz bastante esse trabalho.

Uma alternativa seria oferecer mobilidade acima da camada de rede, o que normalmente acontece com os notebooks de hoje. Quando eles são movidos para novos locais da Internet, adquirem novos endereços de rede. Não existe associação entre os endereços antigos e novos; a rede não sabe que eles pertenciam ao mesmo notebook. Nesse modelo, um notebook pode ser usado para navegar pela Web, mas outros hosts não podem enviar pacotes para ele (por

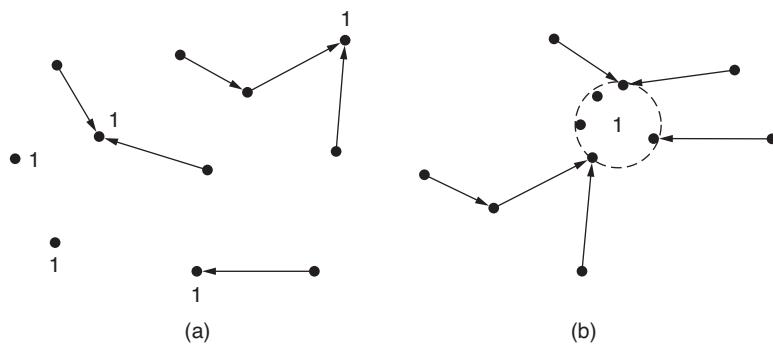



Figura 5.16 | (a) Rotas de anycast para o grupo 1. (b) Topologia vista pelo protocolo de roteamento.

exemplo, para uma chamada que chega), sem montar um serviço local na camada mais alta, por exemplo, acessando o Skype *novamente* depois de mudar de local. Além do mais, as conexões não podem ser mantidas enquanto o host está se movendo; em vez disso, novas conexões precisam ser iniciadas. A mobilidade da camada de rede é útil para resolver esses problemas.

A ideia básica usada para o roteamento móvel na Internet e em redes de celulares é que um host móvel diga a um host no local inicial onde ele está naquele instante. Esse host, que atua em favor do host móvel, é chamado de **agente local**. Quando ele sabe onde o host móvel está localizado no momento, pode encaminhar pacotes de modo que eles sejam entregues.

A Figura 5.17 mostra o roteamento móvel em ação. Um transmissor situado na cidade de Seattle; no noroeste dos Estados Unidos deseja enviar um pacote a um host que normalmente se encontra do outro lado do país, em Nova York. O caso que nos interessa é quando o host móvel não está em seu local. Em vez disso, ele está temporariamente em San Diego.

O host móvel em San Diego precisa adquirir um novo endereço de rede antes que possa usá-la. Isso acontece da maneira normal como os hosts obtêm endereços de rede; veremos como isso funciona para a Internet mais adiante neste capítulo. O endereço local é chamado de **endereço care of**. Quando o host móvel tem esse endereço, ele pode dizer a seu agente local onde ele se encontra naquele momento. Ele faz isso enviando uma mensagem de registro para o agente local (etapa 1) com o endereço care of. A mensagem é mostrada com uma linha tracejada na Figura 5.17 para indicar que se trata de uma mensagem de controle, não uma mensagem de dados.

Em seguida, o transmissor envia um pacote de dados para o host móvel usando seu endereço permanente (etapa 2). Esse pacote é roteado pela rede até o local fixo do host, pois é o lugar do endereço fixo. Em Nova York, o agente local intercepta esse pacote, pois o host móvel está longe de casa. Depois, ele embrulha ou **encapsula** o pacote com um novo cabeçalho e envia esse embrulho para o endereço care of (etapa 3). Esse mecanismo é chamado **tunelamento**. Ele é muito importante na Internet e, por isso, o estudaremos com mais detalhes em outro ponto.

Quando o pacote encapsulado chega ao endereço care of, o host móvel o desembrulha e recupera o pacote do transmissor. O host móvel, em seguida, envia seu pacote de resposta diretamente para o transmissor (etapa 4). A rota final é chamada de **roteamento triangular**, pois pode ser tortuosa se o local remoto estiver longe do inicial. Como parte da etapa 4, o transmissor pode descobrir o endereço care of atual. Os próximos pacotes podem ser roteados diretamente para o host móvel por um túnel até o endereço care of (etapa 5), evitando totalmente o local fixo. Se a conectividade for perdida por algum motivo enquanto o móvel se desloca, o endereço fixo sempre pode ser usado para alcançar o móvel.

Um aspecto importante que omitimos até agora é a segurança. Em geral, quando um host ou roteador recebe uma mensagem da forma "A partir de agora, envie todas as mensagens de correio de Carla para mim", ele pode ter algumas dúvidas sobre quem enviou a mensagem e se fazer isso é uma boa ideia. A informação de segurança está incluída nas mensagens para que sua validade possa ser verificada com protocolos criptográficos, que estudaremos no Capítulo 8.

Existem muitas variações sobre o roteamento móvel. O esquema que apresentamos é modelado com a mobilidade do IPv6, a forma de mobilidade usada na Internet (Johnson et al., 2004) e como parte das redes de celular baseadas em IP, como UMTS. Mostramos o transmissor como um nó fixo para simplificar, mas os projetos permitem que os dois nós sejam hosts móveis. Como alternativa, o host pode fazer parte de uma rede móvel, por exemplo, um computador em um avião. As extensões do esquema básico admitem redes móveis sem nenhum trabalho por parte dos hosts (Devarapalli et al., 2005).

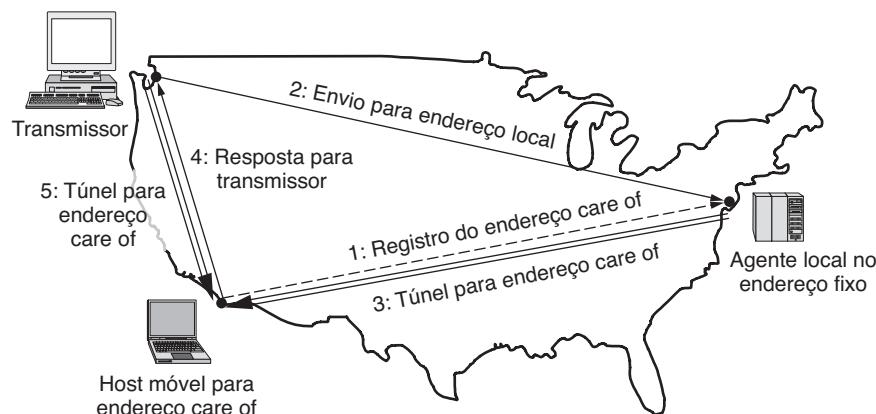



Figura 5.17 | Roteamento de pacotes para hosts móveis.

Alguns esquemas utilizam um agente externo (ou seja, remoto), semelhante ao agente local, mas em um local distante, ou semelhante ao VLR (Visitor Location Register) nas redes de celulares. Porém, nos esquemas mais recentes, o agente externo não é necessário; os hosts móveis atuam como seus próprios agentes externos. De qualquer forma, o conhecimento do local temporário do host móvel é limitado a um pequeno número de hosts (por exemplo, o móvel, o agente local e os transmissores), de modo que os muitos roteadores em uma rede grande não precisam recalcular rotas.

Para obter mais informações sobre roteamento móvel, consulte Perkins (1998, 2002) e Snoeren e Balakrishnan (2000).

5.2.11 Roteamento em redes ad hoc

Vimos como realizar o roteamento quando os hosts são móveis, mas os roteadores são fixos. Um caso ainda mais extremo é aquele em que os próprios roteadores são móveis. Entre as possibilidades estão trabalhadores de emergência no local de um terremoto, veículos militares em um campo de batalha, uma frota de navios no mar ou um grupo de pessoas com notebooks em uma área que não tem instalações 802.11.

Em todos esses casos e em outros, cada nó consiste em um roteador e um host, em geral no mesmo computador. Redes de nós que simplesmente estão próximas entre si são chamadas **redes ad hoc** ou **MANETs** (**Mobile Ad hoc NETworks**). Agora vamos examiná-las rapidamente. Você poderá encontrar mais informações em Perkins (2001).

O que torna as redes ad hoc diferentes das redes fisicamente conectadas é que a topologia é repentinamente abandonada. Os nós podem ir e vir, ou aparecer em novos lugares de um momento para outro. Com uma rede fisicamente conectada, se um roteador tiver um caminho válido para algum destino, esse caminho continuará a ser válido desde que não ocorra uma falha em algum lugar no sistema, o que esperamos ser raro. No caso de uma rede ad hoc, a topologia pode se alterar o tempo todo, e assim o interesse e mesmo a validade dos caminhos podem se alterar de modo espontâneo, sem nenhum aviso. É desnecessário dizer que essas circunstâncias tornam o roteamento em redes ad hoc bem mais desafiador que o roteamento nas redes equivalentes fixas.

Foram propostos diversos algoritmos de roteamento para redes ad hoc. Entretanto, como as redes ad hoc têm sido pouco usadas na prática em comparação com as redes móveis, não sabemos com clareza quais desses protocolos são mais úteis. Como exemplo, vamos examinar um dos mais populares algoritmos de roteamento, o **AODV** (**Ad hoc On-demand Distance Vector**) (Perkins e Royer, 1999). Trata-se de um algoritmo semelhante ao algoritmo de roteamento por vetor de distância, mas adaptado para funcionar em um ambiente móvel, em que os nós geralmente possuem largura de banda limitada e baixa duração das baterias. Agora, vamos ver como ele descobre e mantém as rotas.

Descoberta de rota

No AODV, as rotas para um destino são descobertas por demanda, ou seja, somente quando alguém deseja enviar um pacote para esse destino. Isso economiza muito trabalho, que, de outra forma, seria desperdiçado quando a topologia mudasse antes que a rota fosse usada. Em qualquer instante, a topologia de uma rede ad hoc pode ser descrita por um grafo de nós conectados. Dois nós estão conectados (isto é, têm um arco entre eles no grafo) se podem se comunicar diretamente utilizando seus sinais de rádio. Um modelo básico, porém adequado, suficiente para nossos propósitos, é que cada nó pode se comunicar com todos os outros nós que se encontram dentro de seu círculo de cobertura. As redes reais são mais complicadas, com prédios, morros ou outros obstáculos que bloqueiem sua comunicação, e nós para os quais *A* está conectado com *B*, mas *B* não está conectado com *A*, visto que *A* tem um transmissor mais poderoso que *B*. Contudo, para simplificar, vamos considerar que todas as conexões são simétricas.

Para descrever o algoritmo, considere a rede ad hoc da Figura 5.18, em que um processo no nó *A* deseja enviar um pacote para o nó *I*. O algoritmo AODV mantém uma tabela de vetor de distância em cada nó, classificada por destino, fornecendo informações sobre esse destino, inclusive a que vizinho enviar pacotes para alcançar o destino. Vamos supor que, primeiro, *A* procure em sua tabela e não encontre uma entrada correspondente a *I*. Agora ele tem de descobrir uma rota até *I*. Essa propriedade de descoberta de rotas apenas quando elas são necessárias é o que torna esse algoritmo 'por demanda'.

Para localizar *I*, *A* constrói um pacote ROUTE REQUEST e o transmite por broadcast usando flooding, conforme descrevemos na Seção 5.2.3. A transmissão de *A* alcança *B* e *D*, como ilustra a Figura 5.18(a). Cada nó retransmite a solicitação por broadcast, o que continua até alcançar os nós *F*, *G* e *C* na Figura 5.18(c) e os nós *H*, *E* e *I* na Figura 5.18(d). Um número de sequência definido na origem é usado para eliminar cópias durante o flooding. Por exemplo, *D* descarta a transmissão de *B* na Figura 5.18(c) porque já encaminhou a solicitação.

Por fim, a solicitação alcança *I*, que constrói um pacote ROUTE REPLY. Esse pacote é de unicast para o transmissor, passando pelo caminho inverso seguido pela solicitação. Para que isso funcione, cada nó intermediário precisa se lembrar do nó que lhe enviou a solicitação. As setas na Figura 5.18(b)-(d) mostram a informação de rota inversa armazenada. Cada nó intermediário também incrementa uma contagem de hops enquanto encaminha a resposta. Isso diz aos nós a que distância eles estão do destino. As respostas dizem a cada nó intermediário qual vizinho usar para alcançar o destino: é o nó que lhes enviou a res-

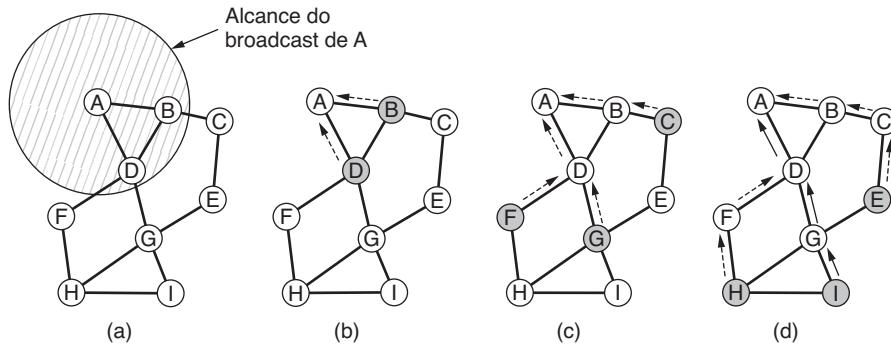



Figura 5.18 | (a) Alcance do broadcast de *A*. (b) Após *B* e *D* receberem o broadcast de *A*. (c) Após *C*, *F* e *G* receberem o broadcast de *A*. (d) Após *E*, *H* e *I* receberem o broadcast de *A*. Os nós sombreados são novos destinatários. As setas mostram as rotas inversas possíveis.

posta. Os nós intermediários *G* e *D* colocam a melhor rota que eles percebem em sua tabela de roteamento enquanto processam a resposta. Quando a resposta alcança *A*, uma nova rota, *ADGI*, terá sido criada.

Em uma rede maior, o algoritmo gera muitos broadcasts, mesmo para destinos que estão próximos. Para reduzir o overhead, o escopo dos broadcasts é limitado, usando o campo TTL (time to live) do pacote IP. Esse campo é iniciado pelo transmissor e decrementado em cada hop. Se ele alcançar 0, o pacote será descartado, em vez de transmitido por broadcast. O processo de descoberta é então modificado como a seguir. Para localizar um destino, o transmissor envia um pacote ROUTE REQUEST por broadcast com o campo TTL definido como 1. Se não houver uma resposta dentro de um período razoável, outro pacote será enviado, dessa vez com TTL definido como 2. As tentativas subsequentes utilizarão 3, 4, 5 etc. Desse modo, a pesquisa será realizada por tentativa, primeiro no local, depois em anéis cada vez mais amplos.

Manutenção de rotas

Por ser possível mover ou desativar os nós, a topologia pode mudar espontaneamente. Por exemplo, na Figura 5.18, se *G* for desativado, *A* não perceberá que a rota que esteve usando para *I* (*ADGI*) não é mais válida. O algoritmo precisa ter a possibilidade de lidar com esse problema. Periodicamente, cada nó transmite por broadcast uma mensagem *Hello*. Cada um de seus vizinhos deve responder a essa mensagem. Se não chegar nenhuma resposta, o transmissor saberá que aquele vizinho saiu de seu alcance e não está mais conectado a ele. De modo semelhante, se ele tentar enviar um pacote a um vizinho que não responde, o nó descobrirá que o vizinho não está mais disponível.

Essas informações são usadas para eliminar rotas que não estão mais ativas. Para cada destino possível, cada nó, *N*, mantém o controle de seus vizinhos que lhe enviaram um pacote para esse destino durante os últimos ΔT segundos. Quando qualquer um dos seus vizinhos se torna inacessível, *N* verifica sua tabela de roteamento para ver quais destinos têm rotas que utilizam o vizinho agora inativo. Para cada uma dessas rotas, cada vizinho ativo é informado de que sua rota que passa por *N* agora é inválida e, portanto, deve ser excluída de sua tabela de roteamento. Em nosso exemplo, *D* elimina suas entradas para *G* e *I* de sua tabela de roteamento e notifica *A*, que elimina sua entrada para *I*. No caso geral, os vizinhos ativos contam para seus vizinhos ativos e assim por diante, recursivamente, até que todas as rotas que dependem do nó que não existe mais sejam eliminadas de todas as tabelas de roteamento.

Nesse estágio, as rotas inválidas foram excluídas da rede e os transmissores podem achar rotas novas e válidas usando o mecanismo de descoberta que descrevemos. Porém, existe uma complicação. Lembre-se de que os protocolos por vetor de distância podem sofrer com uma convergência lenta ou com problemas de contagem ao infinito após uma mudança de topologia, quando eles confundem as rotas antigas e inválidas com as rotas novas e válidas.

Para garantir a convergência rápida, as rotas incluem um número de sequência controlado pelo destino. O número de sequência do destino é como um relógio lógico. O destino o incrementa toda vez que ele envia um novo ROUTE REPLY. Os transmissores pedem uma nova rota incluindo no ROUTE REQUEST o número de sequência de destino da última rota que eles usaram, que será o número de sequência da rota que acabou de ser excluída, ou 0 como um valor inicial. A solicitação será transmitida por broadcast até que uma rota com um número de sequência mais alto seja encontrada. Os nós intermediários armazenam as rotas que possuem um número de sequência mais alto, ou o menor número de hops para o número de sequência atual.

No espírito de um protocolo por demanda, os nós intermediários só armazenam as rotas que estão em uso. Outra informação de rota descoberta durante os broadcasts esgota seu tempo-limite após um pequeno atraso. Descobrir e armazenar apenas as rotas usadas ajuda a economizar largura de banda e tempo de vida da bateria em comparação com um protocolo por vetor de distância padrão, que periodicamente envia atualizações por broadcast.

Até aqui, consideramos apenas uma única rota, de *A* para *I*. Para economizar ainda mais recursos, a descoberta e a manutenção de rotas são compartilhadas quando as rotas se sobrepõem. Por exemplo, se *B* também quiser enviar pacotes para *I*, ele realizará a descoberta de rota. Porém, nesse caso, primeiro a solicitação alcançará *D*, que já tem uma rota para *I*. O nó *D* pode, então, gerar uma resposta para dizer a rota a *B* sem que seja preciso qualquer trabalho adicional.

Existem muitos outros esquemas de roteamento ad hoc. Outro esquema por demanda muito conhecido é o DSR (Dynamic Source Routing) (Johnson et al., 2001). Uma estratégia diferente, baseada na geografia, é explorada pelo GPSR (Greedy Perimeter Stateless Routing) (Karp e Kung, 2000). Se todos os nós souberem suas posições geográficas, o encaminhamento para um destino pode prosseguir sem cálculo de rota, simplesmente apontando na direção correta e circulando de volta para escapar de quaisquer becos sem saída. Saber quais protocolos vencerão dependerá dos tipos de redes ad hoc que forem mais úteis na prática.

5.3 Algoritmos de controle de congestionamento

Quando há pacotes demais presentes em (parte de) uma rede, isso causa atraso de pacotes e uma perda que prejudica o desempenho. Essa situação é chamada **congestionamento**. As camadas de rede e transporte compartilham a responsabilidade de lidar com o congestionamento. Como o congestionamento ocorre dentro da rede, é a camada de rede que o experimenta diretamente e, por fim, precisa determinar o que fazer com os pacotes em excesso. Contudo, o modo mais eficiente de controlar o congestionamento é reduzir a carga que a camada de transporte está colocando sobre a rede. Isso exige que as camadas de rede e de transporte trabalhem juntas. Neste capítulo, examinamos os aspectos do congestionamento da rede. No Capítulo 6, completaremos o assunto abordando os aspectos do congestionamento relativos ao transporte.

A Figura 5.19 ilustra o início do congestionamento. Quando o número de pacotes que os hosts enviam pela rede está dentro de sua capacidade de transporte, o número entregue é proporcional ao número enviado. Se o dobro for enviado, o dobro será entregue. Entretanto, à medida que a carga oferecida se aproxima da capacidade de transporte, rajadas de tráfego ocasionalmente preenchem os buffers dentro dos roteadores e alguns pacotes se perdem. Esses pacotes perdidos consomem parte da capacidade, de modo que o número de pacotes entregues cai para menos do que a curva ideal. A rede agora está congestionada.

A menos que a rede seja bem projetada, ela pode experimentar um **colapso de congestionamento**, em que o desempenho cai enquanto a carga oferecida aumenta além da capacidade. Isso pode acontecer porque os pacotes podem ser atrasados dentro da rede por tanto tempo que não são mais úteis quando saírem dela. Por exemplo, no início da Internet, o tempo que um pacote gastava esperando em um acúmulo de pacotes à sua frente para ser enviado por um enlace lento de 56 kbps atingia o tempo máximo que ele poderia permanecer na rede. Depois disso, ele tinha de ser descartado. Um modo de falha diferente ocorre quando os transmissores retransmitem pacotes já bastante atrasados, achando que eles foram perdidos. Nesse caso, cópias do mesmo pacote serão entregues pela rede, novamente desperdiçando sua capacidade. Para capturar esses fatores, o eixo y da Figura 5.19 é indicado como **goodput**, que é a taxa em que os pacotes *úteis* são entregues pela rede.

Gostaríamos de projetar redes que evitassem o congestionamento sempre que for possível e não sofressem com o colapso se elas se tornarem congestionadas. Infelizmente, o congestionamento não pode ser totalmente evitado. Se os fluxos de pacotes começarem a chegar repentinamente em três ou quatro interfaces de entrada e todos precisarem da mesma interface de saída, uma fila será formada. Se a memória for insuficiente para conter todos eles, os pacotes se perderão. A inclusão de mais memória ajudará até certo ponto, mas Nagle (1987) descobriu que, se os roteadores tiverem um volume infinito de memória, o congestionamento piorará e não melhorará, pois, no momento em que os pacotes chegarem ao início da fila, eles já terão sido temporizados (repetidamente) e as cópias já terão sido enviadas. Isso piora as coisas, e não melhora — leva ao colapso do congestionamento.

Enlaces ou roteadores de pouca largura de banda, que processam pacotes mais lentamente do que a taxa da interface, também podem ficar congestionados. Nesse caso, a situação pode ser melhorada transferindo-se o gargalo para outras partes da rede. Por fim, no entanto, todas as regiões da rede estarão congestionadas. Nessa situação, não existe alternativa além de livrar-se da carga ou montar uma rede mais rápida.

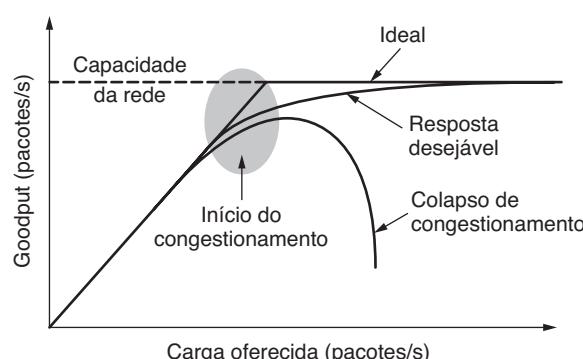



Figura 5.19 | Com muito tráfego, o desempenho cai bruscamente.

Vale a pena destacar a diferença entre controle de congestionamento e controle de fluxo, pois o relacionamento entre eles é muito sutil. O controle de congestionamento se baseia na garantia de que a rede é capaz de transportar o tráfego oferecido. É uma questão global, envolvendo o comportamento de todos os hosts, de todos os roteadores. O controle de fluxo, por outro lado, está relacionado ao tráfego entre um transmissor em particular e um receptor em particular. Sua função é garantir que um transmissor rápido não transmita dados continuamente com mais rapidez do que o receptor é capaz de absorver.

Para perceber a diferença entre esses dois conceitos, considere uma rede de fibra óptica com capacidade de 1.000 Gbps, na qual um supercomputador está tentando transferir um arquivo para um computador pessoal que é capaz de tratar apenas 1 Gbps. Mesmo que não haja congestionamento (a rede em si não apresenta problemas), o controle de fluxo é necessário para forçar o supercomputador a parar com frequência, permitindo que o computador pessoal tenha a chance de 'respirar'.

No outro extremo, considere uma rede com linhas de 1 Mbps e mil computadores de grande porte, metade dos quais está tentando transferir arquivos a 100 kbps para a outra metade. O problema aqui não é o fato de os transmissores rápidos dominarem os receptores lentos, mas sim a questão de o tráfego total oferecido exceder o que a rede é capaz de tratar.

A razão para o controle de congestionamento e o controle de fluxo com frequência serem confundidos é que a melhor forma de lidar com esses dois problemas é fazer com que o host diminua a velocidade. Dessa forma, um host pode receber uma mensagem 'reduzir velocidade', seja porque o receptor não pode manipular a carga, seja porque a rede não é capaz de tratá-la. Voltaremos a esse assunto no Capítulo 6.

Iniciaremos nosso estudo do controle de congestionamento examinando as técnicas que podem ser usadas em diferentes escalas de tempo. Depois, em primeiro lugar veremos estratégias gerais para evitar os congestionamentos, seguidas por técnicas para lidar com os congestionamentos, uma vez que se manifestem.

5.3.1 Técnicas de controle de congestionamento

A presença de congestionamento significa que a carga é (temporariamente) maior do que os recursos (em uma parte da rede) podem tratar. Podemos imaginar duas soluções: aumentar os recursos ou diminuir a carga. Como vemos na Figura 5.20, essas soluções normalmente são aplicadas em diferentes escalas de tempo para impedir o congestionamento ou reagir a ele, quando se manifestar.

A forma mais básica de evitar o congestionamento é criar uma rede que combine bem com o tráfego que ela transporta. Se existe um enlace com pouca largura de banda no caminho ao longo do qual a maior parte do tráfego é direcionada, o congestionamento é provável. Às vezes, recursos podem ser acrescentados dinamicamente quando existe um congestionamento sério, por exemplo, ativando roteadores de reserva ou habilitando linhas que normalmente são usadas apenas como backups (para tornar o sistema tolerante a falhas) ou adquirindo largura de banda no mercado aberto. Mais frequentemente, os enlaces e roteadores normalmente utilizados são atualizados na primeira oportunidade. Isso é chamado **provisionamento**, e acontece em uma escala de tempo de meses, controlada pelas tendências de tráfego a longo prazo.

Para obter o máximo da capacidade da rede existente, as rotas podem ser ajustadas para os padrões de tráfego que mudam durante o dia, à medida que os usuários da rede acordam e dormem em diferentes fusos horários. Por exemplo, rotas podem ser alteradas para deslocar o tráfego para longe de caminhos muito usados alterando os pesos do caminho mais curto. Algumas estações de rádio possuem helicópteros que voam sobre sua cidade para informar sobre congestionamentos nas estradas, possibilitando que aos ouvintes direcionar seus pacotes (carros) para fora dos pontos críticos. Isso é chamado **roteamento com conhecimento do tráfego**. Dividir o tráfego por vários caminhos também é útil.

Entretanto, às vezes não é possível aumentar a capacidade. A única forma de superar o congestionamento é diminuir a carga. Em uma rede de circuito virtual, novas conexões podem ser recusadas se fizerem com que a rede se torne congestionada. Isso é chamado **controle de acesso**.

Em um nível de detalhamento maior, quando o congestionamento é iminente, a rede pode oferecer feedback às fontes cujo fluxo de tráfego é responsável pelo problema. A rede pode solicitar que essas fontes controlem seu tráfego, ou então a própria rede pode atrasar o tráfego.

Duas dificuldades com essa técnica são como identificar o início do congestionamento e como informar à fonte que ela precisa diminuir sua velocidade. Para enfrentar o primeiro problema, os roteadores podem monitorar a carga

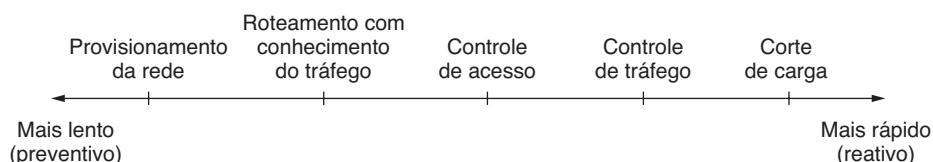



Figura 5.20 | Escalas de tempo das técnicas para o controle de congestionamento.

média, o atraso com enfileiramento ou a perda de pacotes. Em todos os casos, o aumento nos números indica um congestionamento crescente.

Para enfrentar o segundo problema, os roteadores precisam participar de um loop de feedback com as fontes. Para que o esquema funcione de forma correta, a escala de tempo deve ser ajustada cuidadosamente. Se toda vez que dois pacotes chegarem em sequência um roteador gritar PARE, e toda vez que um roteador ficar ocioso por 20 μs gritar SIGA, o sistema oscilará muito e nunca convergirá. Por outro lado, se ele esperar trinta minutos para ter certeza antes de dizer algo, o mecanismo de congestionamento reagirá muito lentamente para ter qualquer uso. A entrega de feedback em tempo é uma questão essencial. Uma questão adicional é fazer com que os roteadores enviem mais mensagens quando a rede já estiver congestionada.

Finalmente, quando tudo o mais falhar, a rede é forçada a descartar pacotes que ela não pode entregar. O nome geral para isso é **corte de carga**. Uma boa política para escolher quais pacotes descartar pode ajudar a impedir o colapso de congestionamento.

5.3.2 Roteamento com conhecimento do tráfego

A primeira técnica que examinaremos é o roteamento com conhecimento do tráfego. Os esquemas de roteamento que examinamos na Seção 5.2 usavam pesos de enlace fixos. Esses esquemas se adaptavam às mudanças na topologia, mas não às mudanças na carga. O objetivo de levar em consideração a carga quando se calculam as rotas é deslocar o tráfego para fora dos pontos críticos, que serão os primeiros lugares na rede a experimentar congestionamento.

O modo mais direto de fazer isso é definir o peso do enlace como uma função da largura de banda do enlace (fixa) e atraso de propagação mais a carga medida (variável) ou atraso médio do enfileiramento. Os caminhos com menor peso, portanto, favorecerão os caminhos menos carregados, se todos os outros critérios forem iguais.

O roteamento com conhecimento do tráfego foi usado no início da Internet de acordo com esse modelo (Khanna e Zinky, 1989). Porém, existe um risco. Considere a rede da Figura 5.21, que é dividida em duas partes, Leste e Oeste, conectadas por dois enlaces, *CF* e *EI*. Suponha que a maior parte do tráfego entre Leste e Oeste esteja usando o enlace *CF* e, como resultado, esse enlace esteja bastante carregado, com longos atrasos. A inclusão do atraso de enfileiramento no peso usado para o cálculo do caminho mais curto tornará *EI* mais atraente. Depois que as novas tabelas de roteamento tiverem sido inseridas, a maior parte do tráfego Leste-Oeste agora passará para *EI*, carregando esse enlace. Consequentemente, na próxima atualização, *CF* parecerá ser o caminho mais curto. Como resultado, as tabelas de roteamento poderão oscilar bastante, ocasionando um roteamento errático e muitos problemas potenciais.

Se a carga for ignorada e apenas a largura de banda e o atraso de propagação forem considerados, esse problema deixará de existir. As tentativas de incluir a carga, mas mudar os pesos dentro de uma faixa estreita, só atrasam as oscilações no roteamento. Duas técnicas podem contribuir para uma solução bem-sucedida. A primeira é o roteamento por caminhos múltiplos, em que pode haver vários caminhos de uma origem a um destino. Em nosso exemplo, isso significa que o tráfego pode ser espalhado pelos dois enlaces, de Leste a Oeste. A segunda é que o esquema de roteamento desloque o tráfego pelas rotas de maneira lenta o suficiente para que seja capaz de convergir, como no esquema de Gallagher (1977).

Com essas dificuldades, na Internet, os protocolos de roteamento geralmente não ajustam suas rotas dependendo da carga. Em vez disso, os ajustes são feitos fora do protocolo de roteamento, alterando lentamente suas entradas. Isso é chamado **engenharia de tráfego**.

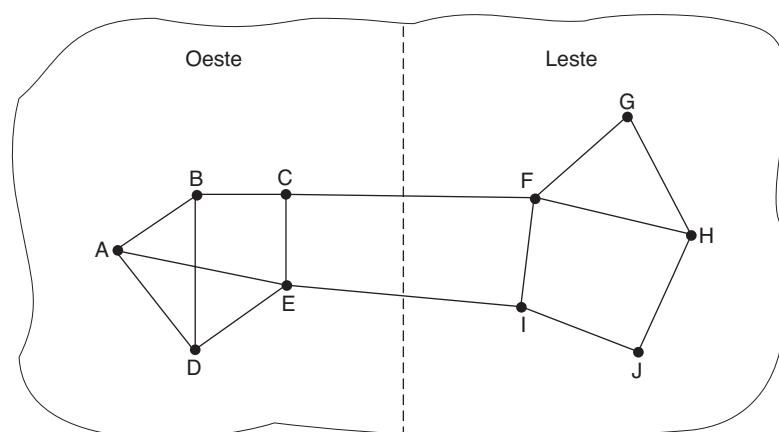



Figura 5.21 | Uma rede em que as partes Leste e Oeste são conectadas por dois enlaces.

5.3.3 Controle de acesso

Uma técnica muito usada em redes de circuito virtual para reduzir o congestionamento é o **controle de acesso**. A ideia é simples: não monte um novo circuito virtual a menos que a rede possa transportar o tráfego adicional sem ficar congestionada. Assim, as tentativas de estabelecer um circuito virtual podem falhar. Isso é melhor do que a alternativa, pois permitir que mais pessoas entrem quando a rede está ocupada só torna as coisas piores. Por analogia, no sistema telefônico, quando uma central está sobrecarregada, ela realiza controle de acesso não emitindo sinais de discar.

O truque com essa técnica é trabalhar quando um novo circuito virtual levar ao congestionamento. A tarefa é simples na rede telefônica, em virtude da largura de banda fixa das chamadas (64 kbps para áudio não compactado). Entretanto, os circuitos virtuais nas redes de computadores têm diversas formas e tamanhos. Assim, o circuito precisa vir com alguma caracterização de seu tráfego se tivermos de aplicar o controle de acesso.

O tráfego normalmente é descrito em termos de sua velocidade e forma. O problema de como descrevê-lo de modo simples, porém significativo, é difícil porque o tráfego é tipicamente feito em rajadas — a velocidade média é apenas metade da história. Por exemplo, é mais difícil lidar com o tráfego que varia enquanto se navega pela Web do que com um streaming de vídeo com o mesmo throughput a longo prazo, pois as rajadas de tráfego da Web têm maior probabilidade de congestionar os roteadores na rede. Um descritor muito utilizado, que captura esse efeito, é o **token bucket** ou **leaky bucket**. Um token bucket tem dois parâmetros que delimitam a velocidade média e o tamanho da rajada instantânea de tráfego. Como os token buckets são muito usados para o quesito qualidade de serviço, vamos examiná-los com detalhes na Seção 5.4.

Armada com as descrições de tráfego, a rede pode decidir se admitirá o novo circuito virtual. Uma possibilidade é que a rede reserve capacidade suficiente ao longo dos caminhos de cada um de seus circuitos virtuais para que o congestionamento não ocorra. Nesse caso, a descrição do tráfego é um acordo de serviço sobre o que a rede garantirá a seus usuários. Impedimos o congestionamento, mas nos desviamos para o tópico relacionado à qualidade de serviço um pouco cedo demais; retornaremos a isso na próxima seção.

Mesmo sem fazer garantias, a rede pode usar descrições de tráfego para controle de acesso. A tarefa é, então, estimar quantos circuitos caberão dentro da capacidade de transporte da rede sem congestionamento. Suponha que todos os circuitos virtuais que possam lidar com o tráfego em velocidades de até 10 Mbps passem pelo mesmo enlace físico de 100 Mbps. Quantos circuitos deverão ser admitidos? É claro que dez circuitos podem ser admitidos sem risco de congestionamento, mas isso é um desperdício no caso normal, pois raramente acontecerá de todos os dez estarem transmitindo em volume máximo ao mesmo tempo. Nas redes reais, medições de comportamento passado, que capturam as estatísticas das transmissões, podem ser usadas para estimar o número de circuitos a admitir, para negociar um melhor desempenho para o risco aceitável.

O controle de acesso também pode ser combinado com o roteamento com conhecimento do tráfego, considerando rotas em torno dos pontos críticos do tráfego como parte do estabelecimento da conexão. Por exemplo, considere a rede ilustrada na Figura 5.22(a), em que duas rotas estão congestionadas.

Suponha que um host conectado ao roteador *A* queira estabelecer uma conexão com um host conectado ao roteador *B*. Normalmente, essa conexão passaria por um dos roteadores congestionados. Para evitar essa situação, podemos redesenhar a rede como mostra a Figura 5.22(b), omitindo os roteadores congestionados e todas as suas interfaces. A linha tracejada mostra uma rota possível para o circuito virtual, que evita os roteadores congestionados. Shaikh et al. (1999) têm um projeto para esse tipo de roteamento sensível à carga.

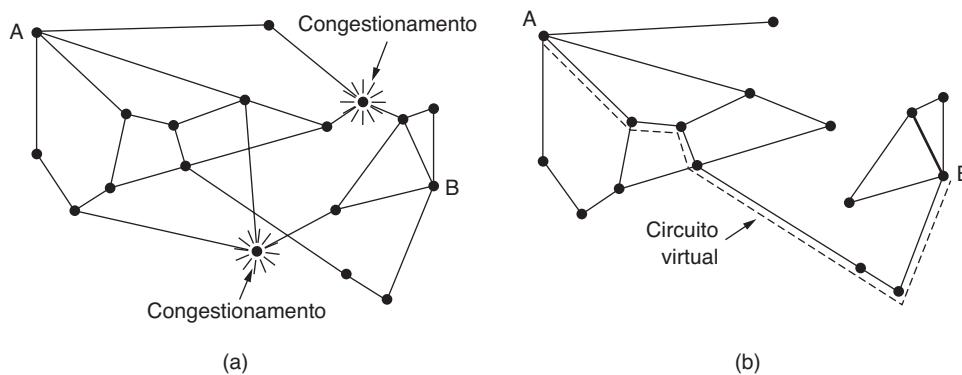



Figura 5.22 | (a) Uma rede congestionada. (b) A parte da rede que não está congestionada. Um circuito virtual de *A* para *B* também aparece.

5.3.4 Controle de tráfego

Na Internet e em muitas outras redes de computadores, os transmissores ajustam suas transmissões para enviar o máximo de tráfego que a rede pode oferecer prontamente. Nesse ambiente, a rede visa a operar imediatamente antes do início do congestionamento. Quando este é iminente, ela precisa dizer aos transmissores para desacelerar suas transmissões e seguir em um ritmo mais lento. Esse feedback é o comportamento normal, não uma situação excepcional. O termo **prevenção de congestionamento** às vezes é usado para comparar esse ponto de operação com um em que a rede se torna (demasiadamente) congestionada.

Agora, vamos examinar algumas técnicas para controlar o tráfego que podem ser usadas nas redes de datagramas e nas redes de circuitos virtuais. Cada uma delas precisa resolver dois problemas. Primeiro, os roteadores precisam determinar quando o congestionamento está se aproximando, de preferência antes que ele seja alcançado. Para fazer isso, cada roteador pode monitorar continuamente os recursos que está usando. Três possibilidades são a utilização dos enlaces de saída, a colocação de pacotes enfileirados no buffer do roteador e o número de pacotes que se perdem em razão do buffering insuficiente. Dessas possibilidades, a segunda é a mais útil. As médias da utilização não são diretamente responsáveis pela explosão da maior parte do tráfego — uma utilização de 50 por cento pode ser baixa para um tráfego tranquilo e muito alta para um tráfego altamente variável. As contagens de pacotes perdidos chegam muito tarde. O congestionamento já terá sido estabelecido quando esses pacotes forem perdidos.

O atraso de enfileiramento dentro dos roteadores captura diretamente qualquer congestionamento experimentado pelos pacotes. Ele quase sempre deve ser baixo, mas saltará quando houver uma rajada de tráfego que gera um acúmulo. Para manter uma boa estimativa do atraso de enfileiramento, d, uma amostra do tamanho instantâneo da fila, s, pode ser tomada periodicamente e d atualizado de acordo com

$$d_{novo} = \alpha d_{antigo} + (1 - \alpha)s$$

onde a constante α determina a velocidade com que o roteador se esquece da história recente. Isso é chamado média móvel ponderada exponencialmente, ou **EWMA** (**Exponentially Weighted Moving Average**). Ela suaviza as flutuações e é equivalente a um filtro passa-baixa. Sempre que d muda para acima do patamar, o roteador observa o início do congestionamento.

O segundo problema é que os roteadores precisam entregar um feedback em tempo aos transmissores que estão causando o congestionamento. Este é experimentado na rede, mas aliviá-lo requer ação por parte dos transmissores que estão usando a rede. Para oferecer feedback, o roteador precisa identificar os transmissores corretos. Ele precisa adverti-los cuidadosamente, sem enviar muito mais pacotes por uma rede já congestionada. Diferentes esquemas utilizam diferentes mecanismos de feedback, conforme descreveremos em seguida.

Pacotes reguladores

O modo mais direto de notificar um transmissor sobre o congestionamento é comunicar-lhe diretamente. Nessa técnica, o roteador seleciona um pacote congestionado e envia um **pacote regulador** de volta ao host de origem, dando-lhe o destino encontrado no pacote. O pacote original pode ser marcado (um bit de cabeçalho é ativado), de modo que não gere mais pacotes reguladores adiante no caminho e seja encaminhado de modo normal. Para evitar o aumento da carga na rede durante um momento de congestionamento, o roteador também pode enviar pacotes reguladores em um ritmo lento.

Quando o host de origem recebe o pacote regulador, ele é solicitado a reduzir o tráfego enviado ao destino especificado, por exemplo, em 50 por cento. Em uma rede de datagramas, simplesmente escolher pacotes aleatórios quando houver um congestionamento provavelmente fará com que os pacotes reguladores sejam enviados a transmissores rápidos, pois eles terão a maioria dos pacotes na fila. O feedback implícito nesse protocolo pode ajudar a impedir o congestionamento, embora não controle nenhum transmissor, a menos que ele cause problema. Pelo mesmo motivo, é provável que vários pacotes reguladores sejam enviados a determinado host de destino. O host deverá ignorar esses pedidos adicionais periodicamente até que sua redução no tráfego tenha surtido efeito. Depois desse período, outros pacotes reguladores indicam que a rede ainda está congestionada.

Um exemplo de pacote regulador usado na antiga Internet é a mensagem SOURCEQUENCH (Postel, 1981). Porém, ela nunca vingou, talvez porque as circunstâncias em que ela foi gerada e o efeito que ela tinha não foram claramente especificados. A Internet moderna usa um projeto de notificação alternativo, que descreveremos a seguir.

Notificação explícita de congestionamento

Em vez de gerar pacotes adicionais para advertir quanto ao congestionamento, um roteador pode marcar qualquer pacote que ele encaminha (definindo um bit no cabeçalho do pacote) para sinalizar que está havendo congestionamento. Quando a rede entrega o pacote, o destino pode observar que existe congestionamento e informar ao transmissor quando ele enviar um pacote de resposta. O transmissor pode, então, reduzir suas transmissões, como antes.

Esse projeto é chamado **notificação explícita de congestionamento**, ou **ECN** (**Explicit Congestion Notification**), e é usado na Internet (Ramakrishnan et al., 2001). Ele é uma melhora de antigos protocolos de sinalização de congestionamento, principalmente o esquema de feedback binário de Ramakrishnan e Jain (1988), que

era usado na arquitetura DECNET. Dois bits no cabeçalho do pacote IP são usados para registrar se o pacote experimentou congestionamento. Os pacotes são desmarcados quando enviados, conforme ilustra a Figura 5.23. Se qualquer um dos roteadores pelos quais eles passarem estiver congestionado, o roteador, então, marcará o pacote como tendo experimentado congestionamento quando ele for encaminhado. O destino então ecoará quaisquer marcas de volta ao transmissor como um sinal explícito de congestionamento em seu próximo pacote de resposta. Isso pode ser visto com uma linha tracejada na figura, para indicar o que acontece acima do nível IP (por exemplo, no TCP). O transmissor precisa, então, reduzir suas transmissões, como no caso dos pacotes reguladores.

Pacotes reguladores hop a hop

Em altas velocidades ou em longas distâncias, muitos pacotes novos podem ser transmitidos após o congestionamento ter sido sinalizado, em razão do atraso, antes que o sinal tenha surtido efeito. Suponha que um host em São Francisco (o roteador *A* da Figura 5.24) esteja enviando tráfego para um host em Nova York (o roteador *D* da Figura 5.24) na velocidade OC-3 de 155 Mbps. Se o host de Nova York começar a esgotar o espaço de buffers, levará cerca de 40 ms para um pacote regulador voltar a São Francisco e solicitar que a transmissão seja mais lenta. Uma indicação de ECN levará ainda mais tempo, pois ela é entregue por meio do destino. A propagação do pacote regulador é mostrada como a segunda, terceira e quarta etapas da Figura 5.24(a). Nesses 40 ms, outros 6,2 megabits terão sido enviados. Mesmo que o host em São Francisco seja imediatamente interrompido, os 6,2 megabits na rede continuarão a trafegar e terão de ser tratados. Somente no sétimo diagrama da Figura 5.24(a) é que o roteador em Nova York notará um fluxo mais lento.

Uma abordagem alternativa é fazer com que o pacote regulador tenha efeito a cada hop pelo qual passar, como mostra a sequência da Figura 5.24(b). Aqui, assim que o pacote regulador atinge *F*, o nó *F* é solicitado a reduzir o fluxo para *D*. Fazendo isso, *F* terá de dedicar mais buffers à conexão, pois a origem ainda estará transmitindo a plena carga, mas dará alívio imediato a *D*, como um remédio para dor de cabeça em um comercial de televisão. Na etapa seguinte, o pacote regulador atingirá *E*, o que o fará reduzir o fluxo para *F*. Essa ação impõe uma demanda maior sobre os buffers de *E*, mas proporciona alívio imediato a *F*. Por fim, o pacote regulador atinge *A* e o fluxo genuinamente diminui sua velocidade.

O efeito líquido desse esquema hop a hop é oferecer alívio rápido no ponto de congestionamento, ao preço de aumentar o consumo de buffers do fluxo ascendente (upstream). Dessa maneira, o congestionamento pode ser cortado pela raiz sem perda de pacotes. A ideia é analisada com mais detalhes em Mishra et al. (1996).

5.3.5 Corte de carga

Quando nenhum dos métodos anteriores fizer o congestionamento desaparecer, os roteadores poderão chamar a artilharia pesada: o corte de carga. **Corte de carga** é uma maneira diferente de dizer que, quando os roteadores estão sendo inundados por pacotes que não podem manipular, eles simplesmente os descartam. A expressão vem do universo da geração de energia elétrica, que se refere à prática das concessionárias de apagar intencionalmente certas áreas para impedir que toda a rede entre em colapso nos dias quentes de verão, quando a demanda por eletricidade ultrapassa muito a capacidade de fornecimento.

A questão principal para um roteador se afogando em pacotes é saber quais pacotes descartar. A escolha preferida pode depender do tipo das aplicações que usam a rede. Para uma transferência de arquivos, um pacote antigo vale mais do que um novo. Isso porque descartar o pacote 6 e manter os pacotes de 7 a 10, por exemplo, só forçará o receptor a realizar mais trabalho para manter em buffer dados que ele ainda não pode usar. Ao contrário, para mídia em tempo real, um novo pacote vale mais do que um antigo. Isso porque os pacotes se tornam inúteis se forem atrasados e perderem o tempo no qual devem ser reproduzidos para o usuário.

A primeira dessas políticas (o antigo é melhor que o novo) costuma ser chamada política do **vinho**, e a segunda (o novo é melhor que o antigo) é chamada política do **leite**, pois a maioria das pessoas preferiria beber leite novo e vinho velho, ao inverso.

O corte de carga mais inteligente exige a cooperação dos transmissores. Um exemplo são os pacotes que transportam informações de roteamento. Esses pacotes são mais importantes do que os pacotes de dados normais, pois estabelecem rotas; se forem perdidos, a rede pode perder a conectividade. Outro exemplo é que certos algoritmos para compactação de vídeo, como MPEG, transmitem periodi-

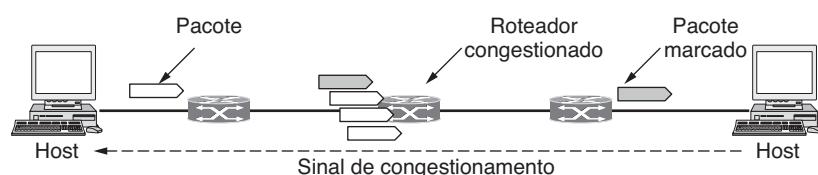



Figura 5.23 | Notificação explícita de congestionamento.

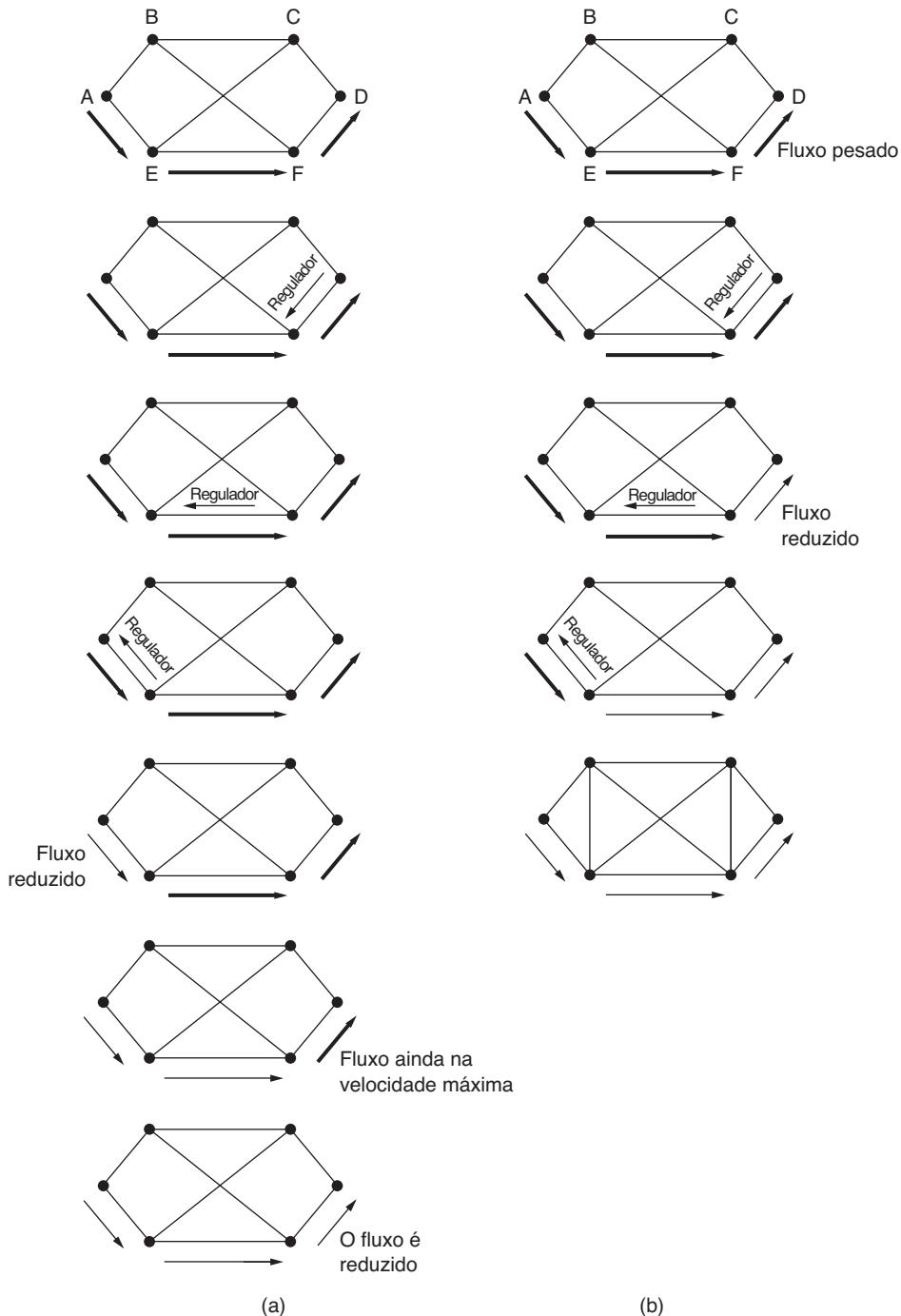



Figura 5.24 | (a) Um pacote regulador que afeta apenas a origem. (b) Um pacote regulador que afeta cada hop pelo qual passa.

camente um quadro inteiro e depois enviam quadros subsequentes sob a forma de diferenças em relação ao último quadro completo. Nesse caso, descartar um pacote que faz parte de uma diferença é preferível a descartar um que faz parte de um quadro completo, pois pacotes futuros dependem do quadro completo.

Para implementar uma política de descarte inteligente, as aplicações devem marcar seus pacotes para indicar à rede qual a importância deles. Depois, quando os pacotes tiverem de ser descartados, os roteadores poderão descartar primeiro os pacotes da classe menos importante, depois os da próxima classe mais importante e assim por diante.

É claro que, a menos que exista algum incentivo especial para evitar marcar os pacotes com MUITO IMPORTANTE — NUNCA DESCARTAR, ninguém o fará. O incentivo pode vir sob a forma de dinheiro, a fim de desencorajar a

marcação inescrupulosa. Por exemplo, a rede pode deixar que os transmissores enviem mais rapidamente do que o permitido pelo serviço que eles adquiriram se marcarem pacotes em excesso como baixa prioridade. Essa estratégia não é realmente uma má ideia, pois faz uso mais eficiente dos recursos ociosos, permitindo que os hosts os utilizem enquanto ninguém mais está interessado, mas sem estabelecer um direito a eles quando a situação fica difícil.

Detecção aleatória prematura

É bem conhecido o fato de que lidar com o congestionamento após sua detecção inicial é mais eficaz do que permitir que o congestionamento se consolide e depois tentar lidar com ele. Essa observação nos leva à ideia de descartar pacotes antes que todo o espaço dos buffers realmente se esgote.

A motivação para essa ideia é que a maioria dos hosts da Internet ainda não recebe sinais de congestionamento dos roteadores na forma da ECN. Em vez disso, a única indicação confiável de congestionamento que os hosts recebem da rede é a perda de pacotes. Afinal, é difícil montar um roteador que não descarte pacotes quando está sobrecarregado. Protocolos de transporte como o TCP são preparados para reagir à perda como congestionamento, tornando a origem mais lenta. O raciocínio por trás dessa lógica é que o TCP foi projetado para redes fisicamente conectadas, as quais são muito confiáveis; assim, a perda de pacotes se deve muito mais ao overflow dos buffers do que a erros de transmissão. Os enlaces sem fios precisam recuperar erros de transmissão na camada de enlace (de modo que não sejam vistos na camada de rede) para que funcionem bem com o TCP.

Esse fato pode ser explorado para ajudar a reduzir o congestionamento. Fazendo os roteadores descartar pacotes antes que a situação se torne desesperadora, a ideia consiste em ter tempo para empreender alguma ação antes que seja tarde demais. Um algoritmo popular para isso é chamado **detecção aleatória prematura**, ou **RED** (**Random Early Detection**) (Floyd e Jacobson, 1993). Para determinar quando começar a descartar, os roteadores mantêm uma média acumulada do tamanho de suas filas. Quando o tamanho médio da fila em algum enlace ultrapassa determinado patamar, o enlace é considerado congestionado e uma pequena fração dos pacotes é descartada aleatoriamente. A escolha aleatória de pacotes torna mais provável que os emissores mais rápidos notem um pacote descartado; essa é a melhor opção, pois o roteador não sabe distinguir qual origem está causando mais problema na rede de datagramas. O transmissor afetado notará a perda quando não houver confirmações e, então, o protocolo de transporte diminuirá a velocidade. Assim, o pacote perdido está entregando a mesma mensagem que o pacote regulador, mas implicitamente, sem o roteador enviar nenhum sinal explícito.

Os roteadores RED melhoram o desempenho em comparação com os roteadores que descartam pacotes quando seus buffers estão cheios, embora possam exigir ajustes para que funcionem bem. Por exemplo, o número ideal de pacotes a descartar depende de quantos transmissores precisam ser notificados do congestionamento. Porém, a ECN é a opção preferida, se estiver disponível. Ela funciona exatamente da mesma maneira, mas entrega um sinal de congestionamento explicitamente, em vez de uma perda; a RED é usada quando os hosts não podem receber sinais explícitos.

5.4 Qualidade de serviço

As técnicas que examinamos nas seções anteriores foram projetadas para reduzir o congestionamento e melhorar o desempenho das redes. Porém, existem aplicações (e clientes) que exigem garantias de desempenho mais altas da rede do que "o melhor que poderia ser feito sob as atuais circunstâncias". Em particular, as aplicações de multimídia frequentemente precisam de um throughput mínimo e latência máxima para funcionar. Nesta seção, continuaremos nosso estudo sobre o desempenho da rede, mas com um foco mais nítido nas alternativas para oferecer uma qualidade de serviço adequada às necessidades das aplicações. Essa é uma área em que a Internet está passando por uma atualização a longo prazo.

Uma solução fácil para fornecer boa qualidade de serviço é montar uma rede com capacidade suficiente para qualquer tráfego que seja jogado nela. O nome para essa solução é **sobreprovisão**. A rede resultante transportará tráfego da aplicação sem perda significativa e, considerando um esquema de roteamento decente, entregará pacotes com baixa latência. O desempenho não fica melhor do que isso. Até certo ponto, o sistema telefônico é sobreprovisionado porque é raro pegar um telefone e não receber um sinal de discar instantaneamente. Simplesmente há muita capacidade disponível para que a demanda sempre seja atendida.

O problema com essa solução é que ela é cara. Ela está basicamente resolvendo um problema jogando dinheiro nele. Os mecanismos de qualidade de serviço permitem que uma rede com menos capacidade atenda aos requisitos da aplicação da mesma forma, mas com menor custo. Além do mais, a sobreprovisão é baseada no tráfego esperado. Todas as apostas estão sobre a mesa se o padrão de tráfego mudar muito. Com mecanismos de qualidade de serviço, a rede pode honrar as garantias de desempenho que ela faz, mesmo quando o tráfego explode, ao custo de reduzir algumas solicitações.

Quatro aspectos devem ser resolvidos para garantir a qualidade de serviço:

1. Que aplicações da rede são necessárias.
2. Como regular o tráfego que entra na rede.

3. Como reservar recursos nos roteadores para garantir o desempenho.
4. Se a rede pode aceitar mais tráfego com segurança.

Nenhuma técnica isolada lida de modo eficaz com todos esses aspectos. Em vez disso, diversas técnicas foram desenvolvidas para ser usadas na camada de rede (e transporte). Soluções práticas para a qualidade de serviço combinam várias técnicas. Para isso, vamos descrever duas versões de qualidade de serviço para a Internet, chamadas Integrated Services e Differentiated Services.

5.4.1 Requisitos da aplicação

Uma sequência de pacotes de uma origem até um destino é chamada **fluxo** (Clark, 1988). Em uma rede orientada a conexões, todos os pacotes que pertencem a um fluxo seguem a mesma rota; em uma rede não orientada a conexões, eles podem seguir rotas diferentes. As necessidades de cada fluxo podem ser caracterizadas por quatro parâmetros principais: largura de banda, atraso, flutuação e perda. Juntos, esses parâmetros determinam a **qualidade de serviço**, ou **QoS** (**Quality of Service**), que o fluxo exige.

Várias aplicações comuns e a rigidez de seus requisitos estão listadas na Tabela 5.2. Observe que os requisitos da rede são menos exigentes do que os da aplicação naqueles casos em que a aplicação pode melhorar o serviço fornecido pela rede. Em particular, as redes não precisam ser isentas de perda para a transferência confiável de arquivos, e elas não precisam entregar pacotes com atrasos idênticos para a reprodução de áudio e vídeo. Alguma perda poderá ser reparada com retransmissões, e alguma quantidade de flutuação pode ser suavizada mantendo-se pacotes em buffer no receptor. Porém, não há nada que as aplicações possam fazer para remediar a situação se a rede oferecer pouca largura de banda ou muito atraso.

As aplicações diferem em suas necessidades em largura de banda, com o e-mail, áudio em todas as suas formas e o login remoto não precisando de muita, mas o compartilhamento de arquivos e vídeo em todas as suas formas precisando de muita largura de banda.

Mais interessantes são os requisitos de atraso. As aplicações de transferência de arquivos, incluindo correio eletrônico e vídeo, não são sensíveis ao atraso. Se todos os pacotes estiverem uniformemente atrasados por alguns segundos, não haverá nenhum dano. Aplicações interativas, como navegação na Web e login remoto, são mais sensíveis ao atraso. Aplicações em tempo real, como telefonia e videoconferência, têm requisitos estritos de atraso. Se todas as palavras em uma ligação telefônica forem atrasadas por um longo tempo, os usuários considerarão a conexão inaceitável. Por outro lado, a reprodução de arquivos de áudio ou vídeo de um servidor não exige baixo atraso.

A variação (ou seja, desvio-padrão) no atraso ou no tempo de chegada do pacote é chamada **flutuação**. As três primeiras aplicações na Tabela 5.2 não são sensíveis aos pacotes que têm entre si intervalos irregulares de chegada. O login remoto às vezes é sensível a isso, pois as atualizações na tela aparecerão em pequenas rajadas se a conexão sofrer muita flutuação. O vídeo e especialmente o áudio são extremamente sensíveis à flutuação. Se o usuário estiver vendo um vídeo pela rede e os quadros forem adiados por exatamente 2,000 segundos, não haverá prejuízo. Mas, se o tempo de transmissão variar aleatoriamente entre 1 e 2 segundos, o resultado será terrível, a menos que a aplicação oculte a flutuação. Para o áudio, uma flutuação de até mesmo alguns milissegundos é claramente perceptível.

As quatro primeiras aplicações têm requisitos mais rigorosos sobre a perda do que áudio e vídeo, pois todos os bits precisam ser entregues corretamente. Esse objetivo normalmente é alcançado com retransmissões de pacotes perdidos na rede pela camada de transporte. Isso é trabalho desperdiçado; seria melhor se a rede recusasse pacotes que ela provavelmente perderia em primeiro lugar. Aplicações de áudio e vídeo podem tolerar alguns pacotes perdidos sem retransmissão, pois as pessoas não observam pausas curtas ou quadros pulados ocasionalmente.

Para acomodar uma série de aplicações, as redes podem dar suporte a diferentes categorias de QoS. Um exem-

Aplicação	Largura de banda	Atraso	Flutuação	Perda
Correio eletrônico	Baixa	Baixo	Baixa	Média
Transferência de arquivos	Alta	Baixo	Baixa	Média
Acesso à Web	Média	Médio	Baixa	Média
Login remoto	Baixa	Médio	Média	Média
Áudio por demanda	Baixa	Baixo	Alta	Baixa
Vídeo por demanda	Alta	Baixa	Alta	Baixa
Telefonia	Baixa	Alta	Alta	Baixa
Videoconferência	Alta	Alta	Alta	Baixa

Tabela 5.2 | Rigidez de requisitos de qualidade de serviço das aplicações.

plo influente vem das redes ATM, que faziam parte de uma grande visão da rede, mas que desde então se tornaram uma tecnologia de nicho. Elas dão suporte a:

1. Taxa de bits constante (por exemplo, telefonia).
2. Taxa de bits variável em tempo real (por exemplo, videoconferência compactada).
3. Taxa de bits variável não de tempo real (por exemplo, assistir a um filme por demanda).
4. Taxa de bits disponível (por exemplo, transferência de arquivos).

Essas categorias também são úteis para outros propósitos e outras redes. A taxa de bits constante é uma tentativa de simular um fio que oferece uma largura de banda uniforme e um atraso uniforme. A taxa de bits variável ocorre quando o vídeo é compactado, com alguns quadros sendo mais compactados que outros. Desse modo, a transmissão de um quadro com uma grande quantidade de detalhes pode exigir o envio de muitos bits, enquanto a transmissão de uma foto de uma parede branca pode se compactar extremamente bem. Os filmes por demanda não são realmente em tempo real, pois alguns segundos de vídeo podem ser facilmente mantidos em buffer no receptor antes que a reprodução seja iniciada, de modo que a flutuação na rede simplesmente causa variação na quantidade de vídeo armazenado, mas não exibido. A taxa de bits disponível se destina a aplicações como correio eletrônico, não sensíveis ao atraso ou à flutuação e que usam a largura de banda que puderem obter.

5.4.2 Modelagem de tráfego

Antes que a rede possa fazer garantias de QoS, ela precisa saber que tráfego está sendo garantido. Na rede telefônica, essa caracterização é simples. Por exemplo, uma chamada de voz (em formato não compactado) precisa de 64 kbps e consiste em uma amostra de 8 bits a cada 125 μs. Porém, o tráfego nas redes de dados é **por rajada**. Ele normalmente chega em taxas não uniformes à medida que a taxa de tráfego varia (por exemplo, videoconferência com compactação), os usuários interagem com as aplicações (por exemplo, navegação em uma nova página Web) e os computadores alternam-se entre as tarefas. As rajadas de tráfego são mais difíceis de lidar do que o tráfego com taxa constante, pois elas podem preencher os buffers e causar perda de pacotes.

A **modelagem de tráfego** é uma técnica relacionada à regulagem da taxa média do fluxo de dados que entra na rede. O objetivo é permitir que as aplicações transmitam uma grande variedade de tráfego que se ajuste a suas necessidades, incluindo algumas rajadas, embora haja um modo simples e útil de descrever os possíveis padrões de tráfego da rede. Quando um fluxo é configurado, o usuário e a rede (isto é, o cliente e o provedor) concordam com determinado padrão de tráfego (ou seja, uma forma) para aquele fluxo. Com efeito, o cliente diz ao provedor "Meu padrão de transmissão se parece com isso; você consegue lidar com ele?".

Às vezes, esse acordo é chamado **acordo de nível de serviço**, ou **SLA** (**Service Level Agreement**), especialmente quando ele é feito por fluxos agregados e longos períodos, como todo o tráfego para determinado cliente. Desde que o cliente cumpra sua parte no negócio e envie somente pacotes que estejam de acordo com o contrato, a concessionária de comunicações promete entregá-los em tempo.

A modelagem de tráfego reduz o congestionamento e ajuda a rede a cumprir sua promessa. Porém, para que isso funcione, também há a questão de como o provedor saberá se o cliente está cumprindo o acordo e o que fazer em caso negativo. Os pacotes que excedem o do padrão combinado podem ser descartados pela rede, ou então podem ser marcados como tendo menor prioridade. O monitoramento de um fluxo de tráfego é chamado **controle de tráfego**.

A modelagem e o controle não são tão importantes para o peer-to-peer e outras transferências que consumirão toda e qualquer largura de banda disponível, mas são de grande importância para dados em tempo real, como conexões de áudio e vídeo, que possuem requisitos rigorosos de qualidade de serviço.

Algoritmos leaky e token bucket

Já vimos um modo de limitar a quantidade de dados que uma aplicação envia: a janela deslizante, que usa um parâmetro para limitar quantos dados estão em trânsito em determinado momento, o que indiretamente limita a velocidade. Agora, veremos um modo mais genérico de caracterizar o tráfego, com os algoritmos leaky bucket e token bucket. As formulações são ligeiramente diferentes, mas dão um resultado equivalente.

Imagine um balde (bucket) com um pequeno furo no fundo, como ilustra a Figura 5.25(b). Independentemente da velocidade com que a água entra no balde, o fluxo de saída ocorrerá a uma taxa constante, R, quando houver qualquer quantidade de água no balde e zero quando o balde estiver vazio. Além disso, quando o balde estiver cheio até a capacidade B, qualquer água que entrar escorrerá pelas bordas e se perderá.

Esse balde pode ser usado para modelar ou controlar os pacotes que entram na rede, como mostra a Figura 5.25(a). Conceitualmente, cada host está conectado à rede por uma interface que contém um leaky bucket. Para enviar um pacote para a rede, deverá ser possível colocar mais água no balde. Se um pacote chegar quando o balde estiver cheio, ou o pacote deverá ser enfileirado até que uma quantidade de água suficiente seja escoada para mantê-lo ou deverá ser descartado. O primeiro caso poderia acontecer em um host modelando seu tráfego para a rede como parte do sistema operacional. O segundo poderia acontecer no hardware, em uma interface de rede do

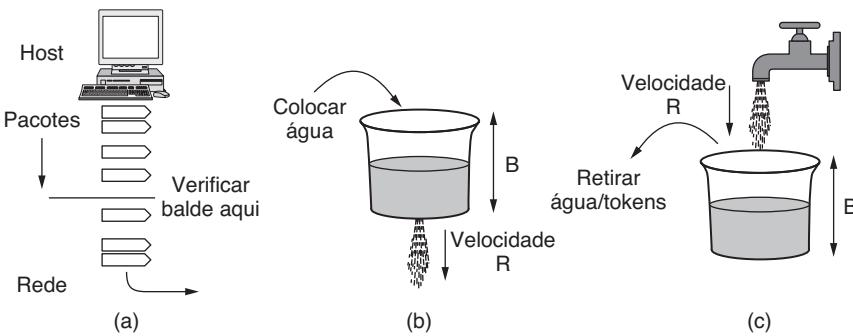



Figura 5.25 | (a) Modelando pacotes. (b) Um leaky bucket. (c) Um token bucket.

servidor que está controlando o tráfego que entra na rede. Essa técnica foi proposta por Turner (1986) e é chamada **algoritmo leaky bucket**.

Uma ideia diferente, porém equivalente, é imaginar a interface de rede como um balde que está sendo enchido, como mostra a Figura 5.25(c). A torneira está jorrando na velocidade *R* e o balde tem uma capacidade *B*, como antes. Agora, para enviar um pacote, temos de poder apanhar água, ou símbolos (tokens), à medida que o conteúdo for chamado normalmente, a partir do balde (em vez de colocar água no balde). Não mais do que um número fixo de tokens, *B*, poderá ficar acumulado no balde, e, se ele estiver vazio, temos de esperar até que mais símbolos cheguem antes de poder enviar outro pacote. Esse algoritmo é chamado **algoritmo token bucket**.

Os leaky e token buckets limitam a velocidade de um fluxo a longo prazo, mas permitem que rajadas de curto prazo até um tamanho máximo regulado passem inalteradas e sem sofrer nenhum atraso artificial. Grandes rajadas serão suavizadas por um modelo de tráfego leaky bucket para reduzir o congestionamento na rede. Como exemplo, imagine que um computador possa produzir dados a 1.000 Mbps (125 milhões de bytes por segundo) e que o primeiro enlace da rede também funcione nessa velocidade. O padrão de tráfego que o host gera aparece na Figura 5.26(a). Esse padrão é em rajadas. A taxa média por segundo é de 200 Mbps, embora o host envie uma rajada de 16.000 KB na velocidade máxima de 1.000 Mbps (por 1/8 de segundo).

Agora, suponha que os roteadores possam aceitar dados na velocidade máxima somente por intervalos curtos, até que seus buffers se encham. O tamanho do buffer é de 9.600 KB, menor que a rajada de tráfego. Para intervalos longos, os roteadores funcionam melhor em velocidades que não ultrapassam 200 Mbps (digamos, porque essa é toda a largura de banda dada ao cliente). A implicação é que, se o tráfego for enviado nesse padrão, parte dele será descartada na rede, pois não cabe nos buffers dos roteadores.

Para evitar essa perda de pacotes, podemos modelar o tráfego no host com um token bucket. Se usarmos uma velocidade, *R*, de 200 Mbps e uma capacidade, *B*, de 9.600

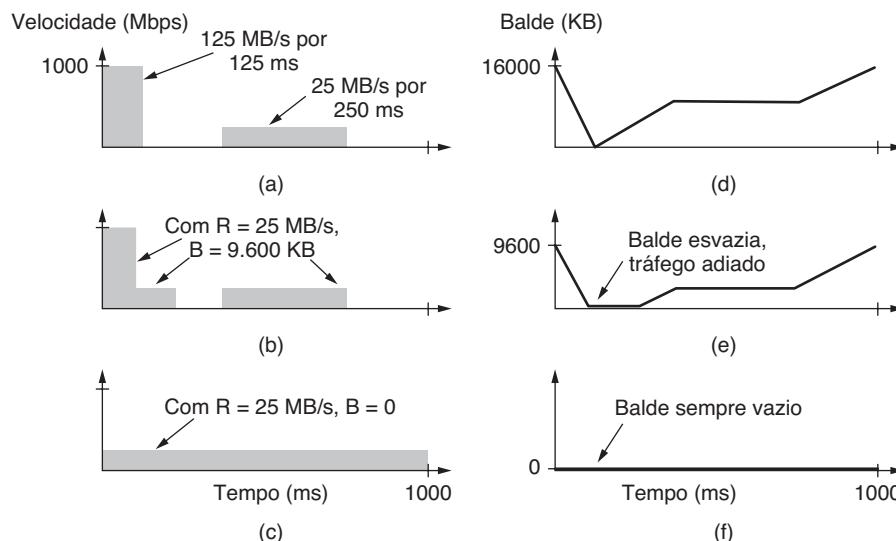



Figura 5.26 | (a) Tráfego de um host. Saída modelada por um token bucket na velocidade de 200 Mbps e capacidade de (b) 9.600 KB e (c) 0 KB. Nível do token bucket para modelagem com velocidade de 200 Mbps e capacidade de (d) 16.000 KB, (e) 9.600 KB e (f) 0 KB.

KB, o tráfego estará dentro daquilo que a rede pode lidar. A saída desse token bucket aparece na Figura 5.26(b). O host pode enviar no máximo a 1.000 Mbps por um curto período até que tenha drenado o balde. Depois, ele precisa recuar para 200 Mbps até que a rajada tenha sido enviada. O efeito é distribuir a rajada com o tempo, pois ela foi muito grande para lidar com tudo ao mesmo tempo. O nível do token bucket aparece na Figura 5.26(e). Ele começa cheio e é esvaziado pela rajada inicial. Quando alcança zero, novos pacotes só podem ser enviados na velocidade em que o buffer está enchendo; não pode haver mais rajadas até que o balde tenha se recuperado. O balde se enche quando nenhum tráfego está sendo enviado e permanece nivelado quando o tráfego está sendo enviado na velocidade do enchimento.

Também podemos modelar o tráfego para que tenha um comportamento menos em rajada. A Figura 5.26(c) mostra a saída de um token bucket com $R = 200$ Mbps e uma capacidade de 0. Esse é o caso extremo em que o tráfego foi completamente suavizado. Nenhuma rajada é permitida, e o tráfego entra na rede em uma velocidade constante. O nível do balde correspondente, mostrado na Figura 5.26(f), sempre está vazio. O tráfego está sendo enfileirado no host para ser lançado na rede e sempre existe um pacote esperando para ser enviado quando for permitido.

Finalmente, a Figura 5.26(d) mostra o nível do balde para um token bucket com $R = 200$ Mbps e uma capacidade de $B = 16.000$ KB. Esse é o menor token bucket através do qual o tráfego passa sem alteração. Ele poderia ser usado em um roteador na rede para controlar o tráfego que o host envia. Se o host estiver enviando tráfego em conformidade com o token bucket, como combinado com a rede, o tráfego passará pelo mesmo token bucket usado no roteador de borda da rede. Se o host enviar em uma velocidade mais rápida, ou com rajada maior, o tráfego não estará em conformidade com o combinado. Ele então removerá os pacotes em excesso ou reduzirá sua prioridade, dependendo do desenho da rede. Em nosso exemplo, o balde se esvazia apenas momentaneamente, no final da rajada inicial, depois recupera o suficiente para a próxima rajada.

Os leaky e token buckets são fáceis de implementar. Agora, vamos descrever a operação de um token bucket. Embora tenhamos descrito a água fluindo continuamente para dentro e fora do balde, as implementações reais precisam funcionar com quantidades discretas. Um token bucket é implementado com um contador para o nível do balde. Esse contador é avançado por $R/\Delta T$ unidades a cada batida de clock de ΔT segundos. Isso seria 200 Kbits a cada ms em nosso exemplo anterior. Toda vez que uma unidade de tráfego é enviada pela rede, o contador é decrementado e o tráfego pode ser enviado até que o contador atinja zero.

Quando todos os pacotes são do mesmo tamanho, o nível do balde pode simplesmente ser contado nos pacotes (por exemplo, 200 Mbits são 20 pacotes de 1.250 bytes). Porém, geralmente são usados pacotes de tamanho variável. Nesse caso, o nível do balde é contado em bytes. Se a contagem de bytes residual é muito baixa para enviar um pacote grande, o pacote precisa esperar até a próxima batida (ou ainda mais tempo, se a taxa de preenchimento for pequena).

Calcular a duração da rajada à taxa máxima (até que o balde esvazie) é um pouco complicado. O cálculo não é feito simplesmente dividindo-se 9.600 KB por 125 MB/s, porque chegam mais símbolos enquanto a rajada está sendo transmitida. Se chamarmos a duração da rajada de S segundos, a taxa de saída máxima de M bytes/s, a capacidade do token bucket de B bytes e a taxa de chegada de símbolos de R bytes/s, podemos ver que uma rajada de saída contém no máximo $B + RS$ bytes. Também sabemos que o número de bytes em uma rajada à velocidade máxima de duração igual a S segundos é MS. Consequentemente, temos:

$$B + RS = MS$$

Podemos resolver essa equação para obter $S = B/(M - R)$. Para nossos parâmetros de $B = 9.600$ KB, $M = 125$ MB/s e $R = 25$ MB/s, obtemos um tempo de rajada de cerca de 94 ms.

Um problema potencial com o algoritmo de token bucket é que ele reduz rajadas grandes para a taxa a longo prazo R. Normalmente desejamos reduzir a taxa de pico, mas sem diminuir a taxa a longo prazo (e também sem aumentar a taxa a longo prazo para permitir mais tráfego na rede). Uma forma de obter um tráfego mais suave é inserir um segundo token bucket após o primeiro. A taxa do segundo balde deve ser muito maior que a do primeiro. Basicamente, o primeiro balde caracteriza o tráfego, consertando sua velocidade média, mas permitindo algumas rajadas. O segundo balde reduz a taxa de pico em que as rajadas são enviadas para a rede. Por exemplo, se a taxa do segundo token bucket for definida como 500 Mbps e a capacidade for definida como 0, a rajada inicial entrará na rede a uma taxa de pico de 500 Mbps, que é menor que a taxa de 1.000 Mbps que tínhamos anteriormente.

O uso desses dois modelos de tráfego pode ser um pouco complicado. Quando os token buckets são usados para a modelagem de tráfego dos hosts, os pacotes são enfileirados e adiados até que os algoritmos permitam que sejam enviados. Quando os token buckets são usados para controle de tráfego nos roteadores da rede, o algoritmo é simulado para garantir que não sejam enviados mais pacotes que o permitido. Não obstante, essas ferramentas fornecem meios para modelar o tráfego de rede de maneira mais gerenciável, a fim de atender aos requisitos de qualidade de serviço.

5.4.3 Listagem de pacotes

A capacidade de regular a forma do tráfego oferecido é um bom início para garantir a qualidade de serviço. Porém, para oferecer uma garantia de desempenho, temos de reservar recursos suficientes ao longo da rota que os pa-

cotes percorrem na rede. Para isso, estamos considerando que todos os pacotes de um fluxo seguem a mesma rota. Dispersar os pacotes pelos roteadores ao acaso torna difícil estabelecer qualquer garantia. Como consequência, algo semelhante a um circuito virtual tem de ser configurado desde a origem até o destino, e todos os pacotes que pertencem ao fluxo devem seguir essa rota.

Os algoritmos que alocam recursos do roteador entre os pacotes de um fluxo e entre fluxos concorrentes são chamados **algoritmos de escalonamento de pacotes**. Três tipos de recursos podem ser reservados para diferentes fluxos:

1. Largura de banda.
2. Espaço em buffer.
3. Ciclos de CPU.

O primeiro recurso, a largura de banda, é o mais óbvio. Se um fluxo exige 1 Mbps e a interface de saída tem uma capacidade de 2 Mbps, tentar orientar três fluxos por essa interface é uma estratégia que não vai funcionar. Desse modo, reservar largura de banda não significa sobrecarregar nenhuma interface de saída.

Um segundo recurso frequentemente escasso é o espaço em buffer. Quando um pacote chega, em geral ele é armazenado no roteador até poder ser transmitido na interface de saída escolhida. A finalidade do buffer é absorver pequenas rajadas de tráfego à medida que os fluxos disputam uns com os outros. Se nenhum buffer estiver disponível, o pacote terá de ser descartado, pois não há lugar para colocá-lo. Para alcançar uma boa qualidade de serviço, alguns buffers podem ser reservados para um fluxo específico, de forma que o fluxo não tenha de competir pelos buffers com outros fluxos. Até um número máximo, sempre haverá um buffer disponível quando o fluxo precisar de um.

Finalmente, os ciclos de CPU também constituem um recurso escasso. O processamento de um pacote exige tempo de CPU do roteador e, assim, o roteador só pode processar certo número de pacotes por segundo. Embora os roteadores modernos sejam capazes de processar a maioria dos pacotes rapidamente, alguns tipos de pacotes exigem maior processamento de CPU, como os pacotes ICMP, que descreveremos na Seção 5.6. É preciso ter certeza de que a CPU não está sobrecarregada, a fim de assegurar o processamento oportuno desses pacotes.

Os algoritmos de escalonamento de pacotes alocam largura de banda e outros recursos do roteador determinando quais dos pacotes no buffer serão enviados na interface de saída em seguida. Já descrevemos o escalonador mais simples quando explicamos o funcionamento dos roteadores. Cada roteador mantém pacotes em buffer em uma fila para cada interface de saída, até que possam ser enviados, e eles são enviados na mesma ordem em que chegaram. Esse algoritmo é conhecido como **FIFO** (**First-In, First-Out**), ou, de modo equivalente, **FCFS** (**First-Come, First-Serve**).

Roteadores FIFO normalmente descartam pacotes recém-chegados quando a fila está cheia. Como o pacote recém-chegado teria sido colocado no final da fila, esse comportamento é chamado **descarte de cauda**. Ele é intuitivo, e você pode estar questionando quais alternativas existem. De fato, o algoritmo RED que descrevemos na Seção 5.3.5 escolheu um pacote recém-chegado para descartar aleatoriamente quando o tamanho médio da fila cresceu muito. Os outros algoritmos de escalonamento que descreveremos também criam outras oportunidades para decidir qual pacote descartar quando os buffers estão cheios.

O escalonamento FIFO é simples de implementar, mas não é adequado para fornecer boa qualidade de serviço, pois, quando existem múltiplos fluxos, um fluxo pode facilmente afetar o desempenho dos outros. Se o primeiro fluxo for agressivo e enviar grandes rajadas de pacotes, eles se alojarão na fila. Pacotes processados na ordem de chegada significam que o transmissor agressivo pode tomar conta da maior parte da capacidade dos roteadores no encaminhamento interno de pacotes, prejudicando os outros fluxos e reduzindo sua qualidade de serviço. Para aumentar o problema, os pacotes de outros fluxos que não são encaminhados internamente provavelmente sofrerão adiamentos, pois eles devem ficar na fila atrás dos muitos pacotes do transmissor mais agressivo.

Muitos algoritmos de escalonamento de pacotes foram criados para oferecer isolamento mais robusto entre os fluxos e afastar tentativas de interferência (Bhatti e Crowcroft, 2000). Um dos primeiros foi o algoritmo de **enfileiramento ordenado com rodízio de filas**, criado por Nagle (1987). A essência desse algoritmo é que os roteadores têm filas separadas, uma para cada fluxo para determinada interface de saída. Quando a interface fica ociosa, o roteador varre as filas em rodízio, como mostra a Figura 5.27. Depois, ele pega o primeiro pacote na próxima fila. Dessa forma, com *n* hosts competindo pela interface de saída, cada host consegue enviar um de cada *n* pacotes. Isso é justo no sentido de que todos os fluxos passam a enviar pacotes na mesma velocidade. O envio de mais pacotes não melhorará essa velocidade.

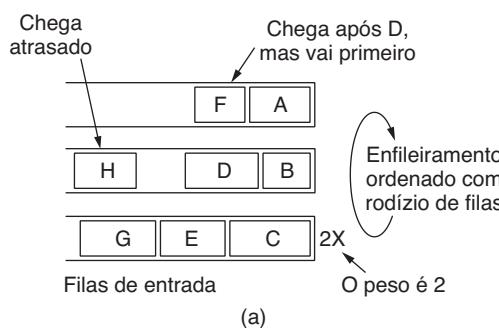
Embora sendo um início, o algoritmo tem uma falha: ele oferece mais largura de banda para os hosts que usam grandes pacotes do que para os que usam pacotes pequenos. Demers et al. (1990) sugeriram uma melhoria em que a operação é feita de modo que simule um rodízio byte a byte, em vez de um rodízio pacote a pacote. O truque é calcular um tempo virtual que seja o número da rodada em que cada pacote acabaria sendo enviado. Cada rodada drena um byte de todas as filas que possuem dados para enviar. Os pacotes são, então, classificados na ordem da hora de finalização e enviados nessa ordem.

Esse algoritmo e um exemplo das horas de finalização para os pacotes que chegam por três fluxos são ilustrados na Figura 5.28. Se um pacote tem tamanho *L*, a rodada em

Figura 5.27 | Enfileiramento ordenado com rodízio de filas.

que ele terminará é simplesmente *L* rodadas após a hora de início. Essa é a hora do término do pacote anterior ou a hora da chegada do pacote, se a fila estiver vazia quando ele chegar.

Pela tabela na Figura 5.28(b) e examinando apenas os dois primeiros pacotes nas duas filas superiores, os pacotes chegam na ordem *A*, *B*, *D* e *F*. O pacote *A* chega na rodada 0 e tem 8 bytes de extensão, de modo que sua hora de término é a rodada 8. De maneira semelhante, a hora de término para o pacote *B* é 11. O pacote *D* chega enquanto *B* está sendo enviado. Sua hora de término é 9 rodadas de byte após iniciar quando *B* chega, ou 20. De modo semelhante, a hora de término para *F* é 16. Na ausência de novas chegadas, a ordem de envio relativa é *A*, *B*, *F*, *D*, embora *F* tenha chegado depois de *D*. É possível que outro pacote pequeno chegue no fluxo superior e obtenha uma hora de término anterior a *D*. Ele só saltará adiante de *D* se a transmissão desse pacote não tiver sido iniciada. O enfileiramento ordenado com rodízio de filas não se apropria de pacotes que estão sendo transmitidos atualmente. Como os pacotes são enviados por inteiro, esse enfileiramento é apenas uma aproximação do esquema byte a byte ideal. Mas ele é uma boa aproximação, ficando dentro de uma transmissão de pacote do esquema ideal em todos os momentos.

Uma deficiência desse algoritmo na prática é que ele dá a todos os hosts a mesma prioridade. Em muitas situações, é desejável dar, por exemplo, mais largura de banda aos servidores de vídeo do que, digamos, aos servidores de arquivos. Isso é facilmente possível dando ao servidor de vídeo dois ou mais bytes por rodada. Esse algoritmo modificado é chamado **enfileiramento ordenado com rodízio de filas ponderado**, ou **WFQ** (**Weighted Fair Queueing**). Deixando o número de bytes por rodada ser o peso de um fluxo, *W*, podemos agora fornecer a fórmula para calcular a hora de término:

$$F_i = \max(A_i, F_{i-1}) + L_i/W$$

onde A_i é a hora de chegada, F_i é a hora de término e L_i é o tamanho do pacote *i*. A fila inferior da Figura 5.28(a) tem um peso 2, de modo que seus pacotes são enviados mais rapidamente, como você pode ver nas horas de término da Figura 5.28(b).

Outra consideração prática é a complexidade da implementação. O WFQ requer que os pacotes sejam inseridos por sua hora de término em uma fila ordenada. Com *N* fluxos, essa é, na melhor das hipóteses, uma operação de ordem $O(\log N)$ por pacote, que é difícil de conseguir para muitos fluxos em roteadores de alta velocidade. Shreedhar e Varghese (1995) descrevem uma aproximação chamada **rodízio por déficit**, que pode ser implementada de modo muito eficiente, com apenas $O(1)$ operações por pacote. O WFQ é muito usado com essa aproximação.

Também existem outros tipos de algoritmos de escalonamento. Um exemplo simples é o escalonamento por prioridade, em que cada pacote é marcado com uma prioridade. Os pacotes com alta prioridade sempre são enviados antes de quaisquer pacotes de baixa prioridade que são mantidos em buffer. Dentro de uma prioridade, os pacotes são enviados na ordem FIFO (ou seja, primeiro a entrar, primeiro a sair). Porém, o escalonamento por prioridade tem a desvantagem de que uma rajada de pacotes de alta prioridade poderá deixar os pacotes de baixa prioridade es-

Pacote	Hora de chegada	Tamanho	Hora de término	Ordem de saída
A	0	8	8	1
B	5	6	11	3
C	5	10	10	2
D	8	9	20	7
E	8	8	14	4
F	10	6	16	5
G	11	10	19	6
H	20	8	28	8

Figura 5.28 | (a) Enfileiramento ordenado com rodízio de filas ponderado. (b) Horas de término para os pacotes.

perando indefinidamente. O WFQ normalmente oferece uma alternativa melhor. Dando à fila de alta prioridade um peso grande, digamos 3, os pacotes de alta prioridade normalmente atravessarão um caminho curto (pois relativamente poucos pacotes devem ter alta prioridade), embora alguma fração dos pacotes de baixa prioridade continue a ser enviada mesmo quando existe tráfego de alta prioridade. Um sistema de prioridade alta e baixa é basicamente um sistema WFQ de duas filas em que a alta prioridade tem peso infinito.

Como um exemplo final de escalonador, os pacotes podem carregar registros de tempo e ser enviados na ordem desses registros. Clark et al. (1992) descrevem um projeto em que o registro de tempo grava quanto o pacote está atrasado ou adiantado enquanto é enviado por uma sequência de roteadores no caminho. Os pacotes que tiverem sido enfileirados atrás de outros em um roteador tendem a estar atrasados, e os pacotes que foram atendidos primeiro tendem a estar adiantados. Enviar pacotes na ordem de registros de tempo tem o efeito benéfico de agilizar pacotes lentos ao mesmo tempo que atrasa pacotes rápidos. O resultado é que todos os pacotes são entregues pela rede com um atraso mais consistente.

5.4.4 Controle de acesso

Agora chegamos ao ponto em que o tráfego de entrada de algum fluxo é bem modelado e pode seguir uma única rota na qual a capacidade pode ser reservada com antecedência nos roteadores ao longo do caminho. Quando tal fluxo é oferecido a um roteador, ele tem de decidir, com base em sua capacidade e na quantidade de compromissos que já assumiu com outros fluxos, se deve admitir ou rejeitar o fluxo.

Já vimos até aqui todos os elementos necessários para a QoS, e agora é hora de reuni-los para realmente oferecê-la. As garantias de QoS são estabelecidas por meio do processo de controle de acesso. Primeiro vimos o controle de acesso usado para regular o congestionamento, o que é uma garantia de desempenho, apesar de fraca. As garantias que estamos considerando agora são mais fortes, mas o modelo é o mesmo. O usuário oferece à rede um fluxo com um requisito de QoS desejado. A rede decide, então, com base em sua capacidade e na quantidade de compromissos que já assumiu para outros fluxos, se deve admitir ou rejeitar o fluxo. Se aceitar, a rede reserva capacidade com antecedência nos roteadores para garantir a QoS quando o tráfego for enviado no novo fluxo.

As reservas devem ser feitas em todos os roteadores ao longo da rota que os pacotes tomarão pela rede. Qualquer roteador no caminho sem reservas poderia se tornar congestionado, e um único roteador congestionado pode quebrar a garantia de QoS. Muitos algoritmos de roteamento encontram o único melhor caminho entre cada origem e cada destino e enviam todo o tráfego pelo melhor caminho. Isso pode fazer com que alguns fluxos sejam rejeitados se não houver capacidade de reserva suficiente ao longo do melhor caminho. As garantias de QoS para novos fluxos ainda podem ser ajustadas escolhendo-se uma rota diferente para o fluxo que tenha capacidade em excesso. Isso é chamado **roteamento por QoS**. Chen e Nahrstedt (1998) oferecem uma visão geral dessas técnicas. Também é possível dividir o tráfego para cada destino por vários caminhos, para encontrar a capacidade em excesso com mais facilidade. Um método simples é que os roteadores escolham caminhos de mesmo custo e dividam o tráfego igual ou proporcionalmente com a capacidade dos enlaces de saída. Porém, algoritmos mais sofisticados também estão disponíveis (Nelakuditi e Zhang, 2002).

Dado um caminho, a decisão de aceitar ou rejeitar um fluxo não é uma simples questão de comparar os recursos (largura de banda, buffers, ciclos) solicitados pelo fluxo com a capacidade em excesso do roteador nessas três dimensões. O problema é um pouco mais complicado. Para começar, embora algumas aplicações possam conhecer seus requisitos de largura de banda, poucas sabem algo sobre buffers ou ciclos de CPU; então, no mínimo, é necessário encontrar uma forma diferente de descrever fluxos e traduzir essa descrição para recursos do roteador. Veremos isso em breve.

Em seguida, algumas aplicações são muito mais tolerantes à perda ocasional dentro de um prazo final que outras. As aplicações precisam escolher entre os tipos de garantias que a rede pode oferecer, sejam elas rígidas, sejam por comportamento que será mantido na maior parte do tempo. Tudo o mais sendo igual, todos gostariam de garantias rígidas, mas a dificuldade é que elas são caras, pois, na pior das hipóteses, restringem o comportamento. As garantias para a maior parte dos pacotes normalmente são suficientes para as aplicações, e mais fluxos com essa garantia oferecida podem ser admitidos como uma capacidade fixa.

Finalmente, algumas aplicações podem estar dispostas a pechinchar sobre os parâmetros de fluxo e outras não. Por exemplo, um aplicativo gerenciador de filmes que normalmente funciona a 30 quadros/s pode estar disposto a reduzir sua velocidade para 25 quadros/s, se não houver largura de banda suficiente para admitir 30 quadros/s. De modo semelhante, o número de pixels por quadro, a largura de banda de áudio e outras propriedades podem ser ajustáveis.

Como muitas partes talvez estejam envolvidas na negociação de fluxo (o transmissor, o receptor e todos os roteadores ao longo do caminho entre eles), os fluxos devem ser descritos com precisão em termos de parâmetros específicos que podem ser negociados. Um conjunto desses parâmetros é chamado **especificação de fluxo**. Em geral, o transmissor (por exemplo, o servidor de vídeo) produz uma especificação de fluxo propondo os parâmetros que

ele gostaria de usar. À medida que a especificação se propaga ao longo da rota, cada roteador a examina e modifica os parâmetros conforme for necessário. As modificações só podem reduzir o fluxo, não aumentá-lo (por exemplo, uma taxa de dados mais baixa, e não mais alta). Quando ela chega à outra extremidade, os parâmetros podem ser estabelecidos.

Como exemplo do que pode haver em uma especificação de fluxo, considere a Tabela 5.3, baseada nas RFCs 2210 e 2211 para os serviços integrados, um projeto de QoS que veremos na próxima seção. Ela tem cinco parâmetros. Os dois primeiros, a *taxa de token bucket* e o *tamanho,* usam um token bucket para fornecer a taxa máxima sustentada que o transmissor pode transmitir, com a média calculada para um longo intervalo de tempo, e a maior rajada que ele pode enviar por um curto intervalo de tempo.

O terceiro parâmetro, a *taxa de dados de pico*, é a taxa máxima de transmissão tolerada, mesmo durante breves intervalos. O transmissor nunca deve exceder essa taxa, mesmo para rajadas curtas.

Os dois últimos parâmetros especificam os tamanhos mínimo e máximo do pacote, incluindo os cabeçalhos da camada de transporte e da camada de rede (por exemplo, TCP e IP). O tamanho mínimo é importante porque o processamento de cada pacote demora um tempo fixo, independentemente de quanto ele seja curto. Um roteador pode estar preparado para manipular 10.000 pacotes/s de 1 KB cada um, mas não estar preparado para tratar 100.000 pacotes/s de 50 bytes cada um, embora isso represente uma taxa de dados mais baixa. O tamanho máximo do pacote é importante em razão de limitações internas da rede que não podem ser excedidas. Por exemplo, se parte do caminho passar por uma rede Ethernet, o tamanho máximo do pacote será restrito a não mais que 1.500 bytes, independentemente do tamanho que o restante da rede possa manipular.

Uma pergunta interessante é como um roteador transforma uma especificação de fluxo em um conjunto de reservas de recursos específicos. À primeira vista, pode parecer que, se um roteador tem um enlace que transmite a, digamos, 1 Gbps e o pacote tem, em média, 1.000 bits, ele pode processar 1 milhão de pacotes/s. Porém, essa observação é falsa, pois sempre haverá períodos ociosos no enlace decorrentes de flutuações estatísticas na carga. Se o enlace precisar de cada bit de capacidade para realizar seu trabalho, a ociosidade até mesmo por alguns bits criará um acúmulo do qual ele nunca poderá se livrar.

Parâmetro	Unidade
Taxa de token bucket	Bytes/s
Tamanho do token bucket	Bytes
Taxa de dados de pico	Bytes/s
Tamanho mínimo do pacote	Bytes
Tamanho máximo do pacote	Bytes

Tabela 5.3 | Um exemplo de especificação de fluxo.

Até mesmo com uma carga ligeiramente abaixo da capacidade teórica, as filas podem se acumular e os atrasos podem ocorrer. Considere uma situação em que os pacotes chegam aleatoriamente com uma taxa de chegada média de λ pacotes/s. Os pacotes possuem tamanhos variáveis e podem ser enviados no enlace com uma taxa de serviço média de μ pacotes/s. Sob a hipótese de que as distribuições de chegada e serviço são distribuições de Poisson (o que é chamado sistema de enfileiramento M/M/1, onde 'M' significa Markov, ou seja, do tipo Poisson), pode ser provado, usando a teoria de enfileiramento, que o atraso médio experimentado por um pacote, T, é

$$T = \frac{1}{\mu} \times \frac{1}{1 - \lambda/\mu} = \frac{1}{\mu} \times \frac{1}{1-\rho}$$

onde $\rho = \lambda/\mu$ é a utilização de CPU. O primeiro fator, $1/\mu$, indica qual seria o tempo do serviço na ausência de competição. O segundo fator é a lentidão ocasionada pela competição com outros fluxos. Por exemplo, se $\lambda = 950.000$ pacotes/s e $\mu = 1.000.000$ pacotes/s, então $\rho = 0{,}95$ e o atraso médio experimentado por pacote será de 20 μs, em vez de 1 μs. Esse tempo considera tanto o tempo de enfileiramento quanto o tempo de serviço, como pode ser visto quando a carga é muito baixa ($\lambda/\mu \approx 0$). Se houver, digamos, trinta roteadores ao longo da rota do fluxo, somente o atraso de enfileiramento será responsável por 600 μs de atraso.

Um método para relacionar especificações de fluxo a recursos do roteador, que corresponde a garantias de largura de banda e desempenho no atraso, é dado por Parekh e Gallagher (1993, 1994). Ele é baseado nas fontes de tráfego modeladas por token buckets (R, B) e WFQ nos roteadores. Cada fluxo recebe um peso de WFQ, W, grande o bastante para drenar sua taxa de token bucket, R, como mostra a Figura 5.29. Por exemplo, se o fluxo tem uma taxa de 1 Mbps e o roteador e o enlace de saída têm uma capacidade de 1 Gbps, o peso para o fluxo precisa ser maior que 1/1.000 do total dos pesos para todos os fluxos nesse roteador para o enlace de saída. Isso garante uma largura de banda mínima para o fluxo. Se ele não puder receber uma taxa grande o bastante, o fluxo não poderá ser admitido.

O maior atraso de enfileiramento que o fluxo verá é uma função do tamanho da rajada do token bucket. Considere os dois casos extremos. Se o tráfego for suave, sem nenhuma rajada, os pacotes serão drenados do roteador tão rapidamente quanto eles chegam. Não haverá atraso de enfileiramento (ignorando os efeitos do uso de pacotes). Por outro lado, se o tráfego for em rajadas, então uma rajada de tamanho máximo, B, poderá chegar ao roteador ao mesmo tempo. Nesse caso, o atraso de enfileiramento máximo, D, será o tempo gasto para drenar essa rajada na largura de

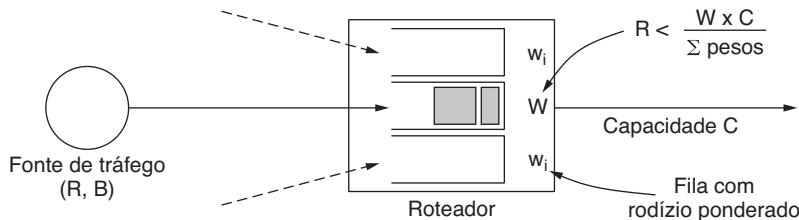



Figura 5.29 | Garantias de largura de banda e atraso com token buckets e WFQ.

banda garantida, ou B/R (novamente ignorando os efeitos do uso de pacotes). Se esse atraso for muito grande, o fluxo deverá solicitar mais largura de banda da rede.

Essas garantias são rígidas. Os token buckets limitam as rajadas da origem, e o enfileiramento ordenado com rodízio de filas isola a largura de banda dada a diferentes fluxos. Isso significa que o fluxo atenderá suas garantias de largura de banda e atraso, independentemente de como os outros fluxos concorrentes se comportam no roteador. Esses outros fluxos não podem quebrar a garantia, mesmo economizando tráfego e com todos enviando ao mesmo tempo.

Além do mais, o resultado é mantido para um caminho por vários roteadores em qualquer topologia de rede. Cada fluxo recebe uma largura de banda mínima, pois essa largura é garantida em cada roteador. O motivo para cada fluxo receber um atraso máximo é mais sutil. Na pior das hipóteses, em que uma rajada de tráfego atinge o primeiro roteador e compete com o tráfego de outros fluxos, ela será adiada até o atraso máximo D. Porém, esse atraso também suavizará a rajada. Por sua vez, isso significa que a rajada não admitirá mais atrasos de enfileiramento em roteadores mais adiante. O atraso de enfileiramento geral será, no máximo, D.

5.4.5 Serviços integrados

Entre 1995 e 1997, a IETF dedicou um grande esforço à criação de uma arquitetura para streaming de multimídia. Esse trabalho resultou em mais de duas dezenas de RFCs, começando com as RFCs 2205 a 2212. O nome genérico desse trabalho é **serviços integrados**. Ele teve como objetivo as aplicações de unicast e multicast. Um exemplo do primeiro tipo de aplicação é um único usuário que recebe um streaming de vídeo transmitido por um site de notícias. Um exemplo do outro tipo de aplicação é um conjunto de estações de televisão digital que transmitem seus programas sob a forma de fluxos de pacotes IP para muitos receptores situados em diversos locais. A seguir, vamos nos concentrar no multicast, pois o unicast é um caso especial de multicast.

Em muitas aplicações de multicast, os grupos podem alterar seus membros dinamicamente; por exemplo, quando as pessoas entram em uma videoconferência ou se entediam e passam para uma novela ou para o canal de esportes. Sob essas condições, a estratégia de fazer com que os transmissores reservem largura de banda com antecedência não funciona muito bem, pois ela exigiria que cada transmissor rastreasse todas as entradas e saídas de sua audiência. No caso de um sistema projetado para transmitir programas de televisão a milhões de assinantes, esse esquema não funcionaria de forma alguma.

RSVP — Resource reSerVation Protocol

A parte principal da arquitetura de serviços integrados visível aos usuários da rede é o **RSVP**, descrito nas RFCs 2205-2210. Esse protocolo é empregado para fazer as reservas; outros protocolos são usados para transmitir os dados. O RSVP permite que vários transmissores enviem os dados para vários grupos de receptores, torna possível aos receptores individuais mudar livremente de canais e otimiza o uso da largura de banda ao mesmo tempo que elimina o congestionamento.

Em sua forma mais simples, o protocolo utiliza roteamento por multicast com spanning trees, como discutido anteriormente. Cada grupo recebe um endereço. Para transmitir dados a um grupo, um transmissor coloca o endereço desse grupo em seus pacotes. Em seguida, o algoritmo de roteamento por multicast-padrão constrói uma spanning tree que cobre todos os membros. O algoritmo de roteamento não faz parte do RSVP. A única diferença em relação ao multicasting normal são algumas informações extras transmitidas periodicamente ao grupo por multicast, a fim de informar aos roteadores ao longo da árvore que devem manter certas estruturas de dados em suas respectivas memórias.

Como exemplo, considere a rede da Figura 5.30(a). Os hosts 1 e 2 são transmissores de multicast, e os hosts 3, 4 e 5 são receptores de multicast. Nesse exemplo, os transmissores e os receptores estão separados, mas, em geral, os dois conjuntos podem se sobrepor. As árvores de multicast para os hosts 1 e 2 são mostradas na Figura 5.30(b) e na Figura 5.30(c), respectivamente.

Para obter uma melhor recepção e eliminar o congestionamento, qualquer um dos receptores de um grupo pode enviar uma mensagem de reserva pela árvore para o transmissor. A mensagem é propagada com a utilização do algoritmo de encaminhamento pelo caminho inverso, discutido antes. Em cada hop, o roteador detecta a reserva e guarda a largura de banda necessária. Na seção anterior,

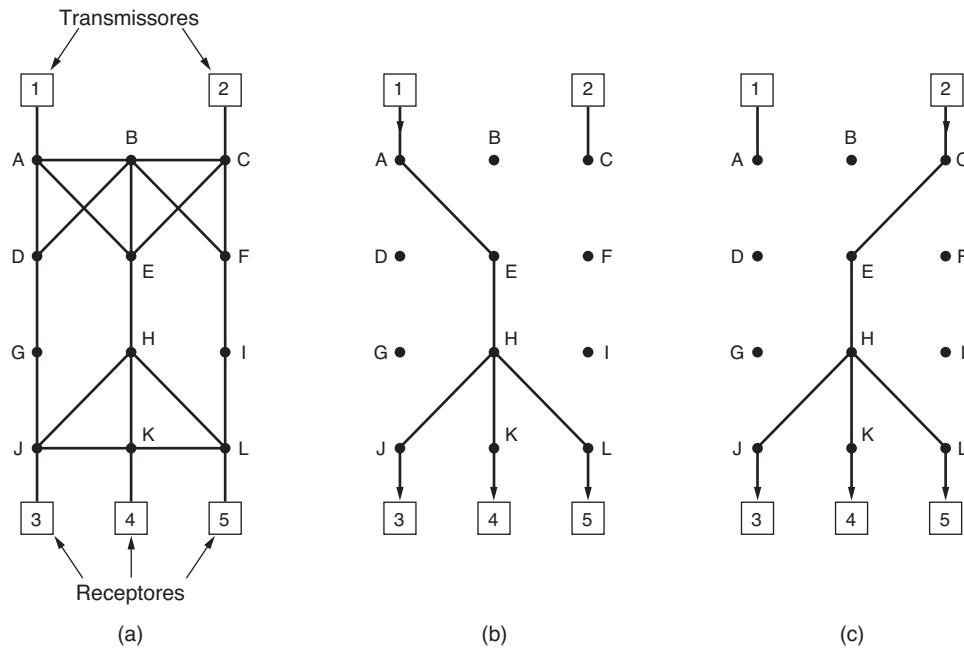



Figura 5.30 | (a) A rede. (b) A spanning tree de multicast para o host 1. (c) A spanning tree de multicast para o host 2.

vimos como um escalonador de enfileiramento ordenado com rodízio de filas ponderado pode ser usado para fazer essa reserva. Se a largura de banda disponível não for suficiente, ele informa a falha. No momento em que a mensagem retornar à origem, a largura de banda já terá sido reservada ao longo de todo o caminho entre o transmissor e o receptor, fazendo a solicitação de reserva ao longo da spanning tree.

Um exemplo desse processo de reserva pode ser visto na Figura 5.31(a). Aqui, o host 3 solicitou um canal ao host 1. Uma vez estabelecido o canal, os pacotes podem fluir do host 1 até o host 3, sem congestionamento. Agora, considere o que acontecerá se, em seguida, o host 3 reservar um canal para o outro transmissor, o host 2, de forma que o usuário possa assistir a dois programas de televisão ao mesmo tempo. Um segundo caminho será reservado, como ilustra a Figura 5.31(b). Observe que são necessários dois canais distintos entre o host 3 e o roteador *E*, pois dois fluxos independentes estão sendo transmitidos.

Por fim, na Figura 5.31(c), o host 5 decide assistir ao programa que está sendo transmitido pelo host 1 e também faz uma reserva. Primeiro, é reservada uma largura

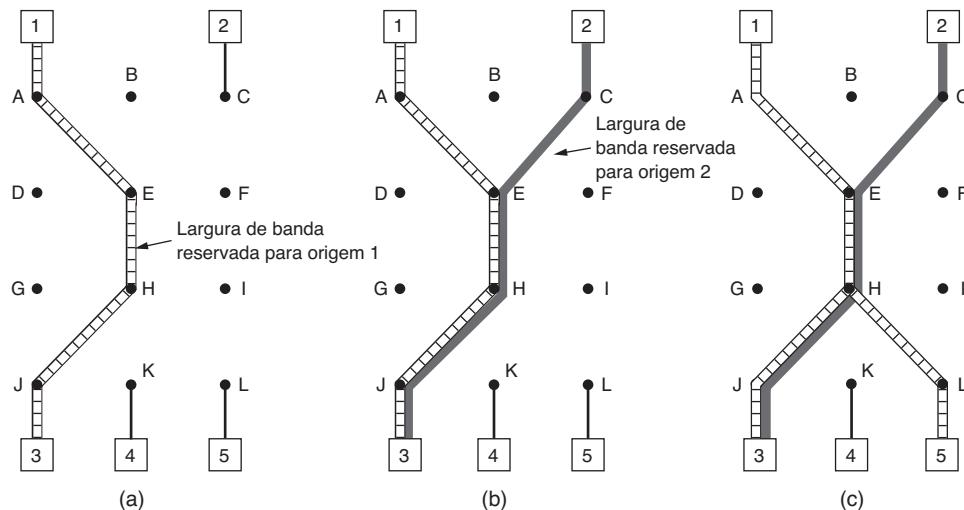



Figura 5.31 | (a) O host 3 solicita um canal ao host 1. (b) Em seguida, o host 3 solicita um segundo canal, agora ao host 2. (c) O host 5 solicita um canal ao host 1.

de banda dedicada até o roteador *H*. Entretanto, esse roteador percebe que já está sendo alimentado pelo host 1; assim, como a largura de banda necessária já foi reservada, ele não precisa reservar mais nada. Observe que os hosts 3 e 5 poderiam ter solicitado diferentes volumes de largura de banda (por exemplo, o host 3 tem uma tela pequena e só deseja a informação em baixa resolução); portanto, a capacidade reservada deve ser grande o suficiente para satisfazer ao receptor mais voraz.

Ao fazer uma reserva, um receptor pode (opcionalmente) especificar uma ou mais origens a partir das quais deseja receber informações. Ele também pode especificar se essas opções serão fixas durante o período de reserva, ou se o receptor deseja manter em aberto a opção de alterar as origens mais tarde. Os roteadores utilizam essas informações para otimizar o planejamento da largura de banda. Em particular, dois receptores só são configurados para compartilhar um caminho se ambos concordarem em não alterar as origens posteriormente.

O motivo para essa estratégia no caso totalmente dinâmico é que a largura de banda reservada é desacoplada da opção de origem. Quando reserva a largura de banda, um receptor pode alternar para outra origem e manter a parte do caminho existente que for válida para a nova origem. Por exemplo, se o host 2 estiver transmitindo diversos fluxos de vídeo, o host 3 poderá alternar entre eles quando quiser sem alterar sua reserva, pois os roteadores não se importam com o programa a que o receptor está assistindo.

5.4.6 Serviços diferenciados

Os algoritmos baseados em fluxo têm potencial para oferecer boa qualidade de serviço a um ou mais fluxos, porque eles reservam quaisquer recursos necessários ao longo da rota. Porém, eles também têm a desvantagem de exigir uma configuração antecipada para estabelecer cada fluxo, algo que não se ajusta bem quando existem milhares ou milhões de fluxos. Além disso, eles mantêm o estado interno por fluxo nos roteadores, tornando-os vulneráveis a quedas. Por fim, as mudanças exigidas no código do roteador são substanciais e envolvem trocas complexas de roteador para roteador, a fim de configurar os fluxos. Em consequência disso, ainda existem poucas implementações de RSVP ou de algo semelhante.

Por essas razões, a IETF também criou uma abordagem mais simples para oferecer qualidade de serviço, uma estratégia que pode ser implementada em grande parte no local em cada roteador, sem configuração antecipada e sem ter de envolver todo o caminho. Essa abordagem é conhecida como qualidade de serviço **baseada em classe** (em vez de baseada em fluxo). A IETF padronizou uma arquitetura para ela, chamada arquitetura de **serviços diferenciados**, descrita nas RFCs 2474, 2475 e várias outras. Vamos descrevê-la agora.

Os serviços diferenciados (Differentiated Services — DS) podem ser oferecidos por um conjunto de roteadores que formam um domínio administrativo (por exemplo, um ISP ou uma empresa de telecomunicações). A administração define um conjunto de classes de serviço com regras de encaminhamento correspondentes. Se um cliente fizer a assinatura para DS, seus pacotes que entrarem no domínio serão marcados com a classe a que pertencem. Essa informação é executada no campo *differentiated services* dos pacotes IPv4 e IPv6 (descritos na Seção 5.6). As classes são definidas como **comportamentos por hop**, pois correspondem ao tratamento que o pacote receberá em cada roteador, e não uma garantia pela rede. Um serviço melhor é fornecido aos pacotes com alguns comportamentos por hop (por exemplo, um serviço especial) e não a outros (por exemplo, o serviço regular). O tráfego dentro de uma classe talvez tenha de obedecer a alguma forma específica, como a de um leaky bucket com uma taxa de escoamento especificada. Uma operadora com certo tino comercial poderia cobrar uma tarifa extra por cada pacote especial transportado, ou poderia permitir até *N* pacotes especiais por mês a uma taxa mensal adicional fixa. Observe que esse esquema não exige nenhuma configuração antecipada, nenhuma reserva de recursos e nenhuma negociação demorada e ponto a ponto para cada fluxo, como ocorre no caso dos serviços integrados. Isso torna relativamente fácil implementar os serviços diferenciados.

O serviço baseado em classe também ocorre em outros campos. Por exemplo, as empresas de entrega de pacotes frequentemente oferecem serviço noturno, em dois e em três dias. As empresas aéreas oferecem serviço de primeira classe, classe executiva e classe econômica. Os trens interurbanos muitas vezes têm várias classes de serviço. Até mesmo o metrô de Paris tem duas classes de serviço. No caso dos pacotes, as classes de serviço podem diferir em termos de atraso, flutuação e probabilidade de os pacotes serem descartados na eventualidade de ocorrer congestionamento, entre outras possibilidades (mas talvez não de quadros Ethernet, mais espaçosos).

Para tornar mais clara a diferença entre a qualidade de serviço baseada em fluxo e a qualidade de serviço baseada em classe, vamos considerar o exemplo da telefonia na Internet. Com um esquema baseado em fluxo, cada chamada telefônica obtém seus próprios recursos e garantias. Com um esquema baseado em classe, todas as chamadas telefônicas juntas obtêm os recursos reservados para a classe de telefonia. Esses recursos não podem ser tirados pelos pacotes da classe de navegação Web ou de outras classes, mas nenhuma chamada telefônica recebe qualquer recurso privado reservado apenas para ela.

Encaminhamento expresso

A escolha de classes de serviço cabe a cada operadora, mas, como os pacotes com frequência são encaminhados entre redes pertencentes a diferentes operadoras, a IETF definiu algumas classes de serviço independentes da rede.

A mais simples dessas classes é a de **encaminhamento expresso**; portanto, vamos começar por ela. Essa classe é descrita na RFC 3246.

A ideia que rege o encaminhamento expresso é muito simples. Há duas classes de serviço disponíveis: regular e expressa. A grande maioria do tráfego deve ser regular, mas uma pequena fração dos pacotes é expressa. Os pacotes expressos devem ser capazes de transitar pela rede como se nenhum outro pacote estivesse presente. Desse modo, eles receberão serviço com poucas perdas, atrasos e flutuações — exatamente o que é necessário para o VoIP. Uma representação simbólica desse sistema de 'dois tubos' é dada na Figura 5.32. Observe que ainda existe apenas uma linha física. Os dois canais lógicos mostrados na figura representam um modo de reservar largura de banda, não uma segunda linha física.

Um modo de implementar essa estratégia é o seguinte. Os pacotes são classificados como expressos ou regulares e marcados de acordo com esse critério. Essa etapa pode ser feita no host transmissor ou no (primeiro) roteador de ingresso. A vantagem de fazer a classificação no host transmissor é que mais informações estão disponíveis a respeito de quais pacotes pertencem a quais fluxos. Essa tarefa pode ser realizada pelo software de rede ou ainda pelo sistema operacional, para evitar ter de mudar as aplicações existentes. Por exemplo, está se tornando comum que os pacotes VoIP sejam marcados para serviço expresso pelos hosts. Se os pacotes passarem por uma rede corporativa ou ISP que aceite o serviço expresso, eles receberão tratamento preferencial. Se a rede não aceitar o serviço expresso, nenhum dano será causado.

Naturalmente, se a marcação for feita pelo host, o roteador de ingresso provavelmente policiará o tráfego para garantir que os clientes não estejam enviando mais tráfego expresso do que aqueles que efetivamente pagaram. Dentro da rede, os roteadores podem ter duas filas de saída para cada interface de saída, uma para os pacotes expressos e outra para os pacotes regulares. Quando um pacote chega, ele é enfileirado de acordo com seu tipo. A fila expressa recebe mais prioridade que a regular, por exemplo, usando um escalonador de prioridade. Desse modo, os pacotes expressos veem uma rede desafogada, mesmo quando, na realidade, existe uma alta carga de tráfego regular.

Encaminhamento garantido

Um esquema um pouco mais elaborado para gerenciar as classes de serviço é chamado **encaminhamento garantido**. Ele é descrito na RFC 2597 e especifica que haverá quatro classes de prioridade, cada uma delas tendo seus próprios recursos. As três classes superiores poderiam ser chamadas ouro, prata e bronze. Além disso, ele define três classes de descarte de pacotes que estejam sofrendo congestionamento: baixo, médio e alto. Considerados em conjunto, esses dois fatores definem 12 classes de serviço.

A Figura 5.33 mostra uma forma possível de processar pacotes no esquema de encaminhamento garantido. A primeira etapa consiste em classificar os pacotes em uma das quatro classes de prioridade. Como antes, essa etapa poderia ser feita no host transmissor (como mostra a figura) ou no roteador de ingresso, e os pacotes de prioridade mais alta podem ser limitados pelo operador como parte da oferta de serviço.

A próxima etapa é determinar a classe de descarte para cada pacote. Isso é feito pela passagem dos pacotes de cada classe de prioridade por um gerenciador de tráfego, como um token bucket. O gerenciador permite que todo o tráfego passe, mas identifica os pacotes que cabem dentro de pequenas rajadas como baixo descarte, pacotes que excedem as pequenas rajadas como descarte médio e pacotes que ultrapassam grandes rajadas como descarte alto. A combinação de classes de prioridade e descarte é, então, codificada em cada pacote.

Finalmente, os pacotes são processados pelos roteadores na rede com um escalonador de pacotes que distingue as diferentes classes. Uma escolha comum é usar o enfileiramento ordenado com rodízio de filas ponderado para as quatro classes de prioridade, com as classes mais altas recebendo pesos maiores. Desse modo, as classes mais altas receberão mais largura de banda, mas as classes mais baixas não ficarão totalmente sem largura de banda. Por exemplo, se os pesos dobrarem de uma classe para a próxima, mais alta, a classe mais alta receberá o dobro da largura de banda. Dentro de uma classe de prioridade, os pacotes com uma classe de descarte mais alta podem ser preferencialmente descartados executando-se um algoritmo como RED (Random Early Detection), que vimos na Seção 5.3.5. O

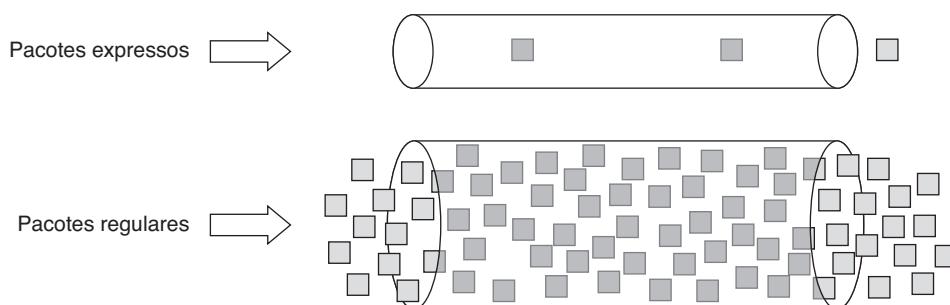



Figura 5.32 | Pacotes expressos experimentam uma rede sem tráfego.

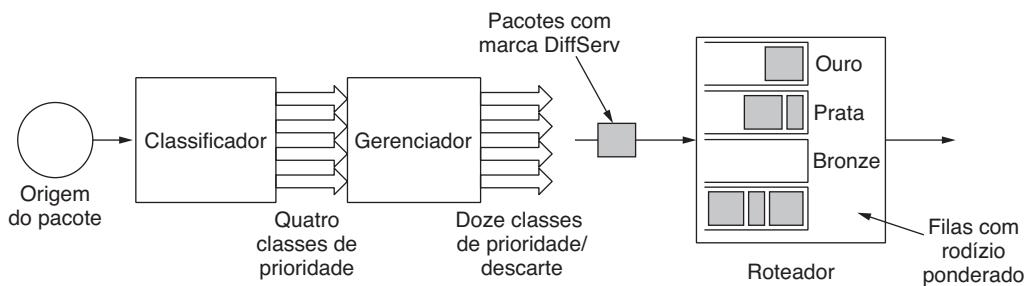



Figura 5.33 | Uma implementação possível do fluxo de dados para encaminhamento garantido.

RED começará a descartar pacotes à medida que o congestionamento se acumula, mas antes que o roteador tenha ficado sem espaço em buffer. Nesse estágio, ainda existe espaço em buffer para aceitar pacotes com baixo descarte enquanto pacotes com alto descarte são removidos.

5.5 Interligação de redes

Até agora, supusemos implicitamente que havia uma única rede homogênea, com cada máquina usando o mesmo protocolo em cada camada. Infelizmente, essa suposição é muito otimista. Existem muitas redes diferentes, incluindo PANs, LANs, MANs e WANs. Descrevemos Ethernet, Internet por cabo, as redes de telefone fixo e móvel, 802.11, 802.16 e outras. Diversos protocolos estão sendo bastante utilizados em cada camada dessas redes. Nas seções a seguir, examinaremos cuidadosamente as questões que surgem quando duas ou mais redes são interconectadas para formar uma **rede interligada**, ou, simplesmente, uma **internet** (com 'i' minúsculo).

Seria muito mais simples juntar as redes se todos usassem uma única tecnologia de rede, e normalmente acontece de existir um tipo dominante de rede, como a Ethernet. Não sabemos se essa multiplicidade de tecnologias é uma condição temporária, que passará tão logo alguém perceba quanto a rede [preencha com sua rede favorita] é maravilhosa. Mas não conte com isso. A história mostra que esse é um pensamento de algo que se deseja. Diferentes tipos de redes resolvem diferentes problemas, de modo que, por exemplo, a Ethernet e as redes por satélite provavelmente serão sempre diferentes. A reutilização de sistemas existentes, como as redes por cabo, a rede telefônica e as linhas da rede de energia elétrica, acrescenta restrições que causam divergências nos recursos da rede. A heterogeneidade veio para ficar.

Se sempre haverá redes diferentes, seria mais simples se não precisássemos interconectá-las. Isso também é pouco provável. Bob Metcalfe postulou que o valor de uma rede com N nós é o número de conexões que podem ser feitas entre os nós, ou N^2 (Gilder, 1993). Isso significa que grandes redes são muito mais valiosas do que as redes pequenas, pois elas permitem muito mais conexões, de modo que sempre haverá um incentivo para combinar redes menores.

A Internet é o principal exemplo dessa interconexão. (Escreveremos Internet com a letra "I" maiúscula para distingui-la de outras internets, ou redes interligadas.) A finalidade de juntar todas essas redes é permitir que os usuários em qualquer uma delas se comuniquem com os usuários em todas as outras. Quando você paga a um ISP pelo serviço da Internet, o valor pode ser cobrado de acordo com a largura de banda de sua conexão, mas o que você está realmente pagando é a capacidade de trocar pacotes com qualquer outro host que também esteja conectado à Internet. Afinal, a Internet não seria muito popular se você só pudesse enviar pacotes para outros hosts na mesma cidade.

Como as redes interligadas normalmente diferem em aspectos importantes, levar pacotes de uma rede para outra nem sempre é tão fácil. Temos de resolver problemas de heterogeneidade e também problemas de escala enquanto a rede interligada resultante fica maior. Vamos começar examinando como as redes podem diferir para ver com que estamos lidando. Depois, veremos a técnica usada com tanto sucesso pelo IP (Internet Protocol), o protocolo da camada de rede da Internet, incluindo técnicas para tunelamento em redes, roteamento em redes interligadas e fragmentação de pacotes.

5.5.1 Diferenças entre redes

As redes podem diferir em várias aspectos. Algumas dessas diferenças, como técnicas de modulação ou formatos de quadros distintos, encontram-se nas camadas física e de enlace de dados. Essas diferenças não nos interessam agora. Em vez disso, na Tabela 5.4 listamos algumas diferenças que podem ocorrer na camada de rede. É a superação dessas diferenças que torna a interligação de redes mais difícil do que a operação de uma única rede.

Quando os pacotes enviados por uma origem em uma rede devem transitar por uma ou mais redes externas antes de chegar à rede de destino, podem ocorrer muitos problemas nas interfaces existentes entre as redes. Para começar, a origem precisa ser capaz de endereçar o destino. O que fazemos quando uma origem está em uma rede Ethernet

Item	Algumas possibilidades
Serviço oferecido	Orientado a conexões e não orientado a conexões
Endereçamento	Diferentes tamanhos, simples ou hierárquico
Broadcasting	Presente ou ausente (também multicast)
Tamanho do pacote	Cada rede tem seu próprio tamanho máximo
Ordenação	Entrega ordenada e não ordenada
Qualidade de serviço	Pode estar presente ou ausente; muitos tipos
Confiabilidade	Diferentes níveis de perda
Segurança	Regras de privacidade, criptografia etc.
Parâmetros	Diferentes timeouts, especificações de fluxo etc.
Contabilidade	Por tempo de conexão, por pacote, por byte ou nenhuma

Tabela 5.4 | Algumas das muitas diferenças possíveis entre redes.

e o destino está em uma rede WiMAX? Supondo que ainda possamos especificar um destino WiMAX a partir de uma rede Ethernet, os pacotes cruzariam de uma rede não orientada a conexões para uma rede orientada a conexões. Isso pode exigir que uma nova conexão seja configurada sem aviso prévio, o que introduz atraso, e muito overhead se a conexão não for usada para muito mais pacotes.

Muitas diferenças específicas também podem ter de ser acomodadas. Como realizamos o multicast de um pacote a um grupo com alguns membros em uma rede que não admite multicast? Os diferentes tamanhos máximos de pacotes usados por redes distintas também são uma grande dor de cabeça. Como passar um pacote de 8.000 bytes por uma rede cujo tamanho máximo é de 1.500 bytes? Se os pacotes em uma rede orientada a conexões transitarem por uma rede não orientada a conexões, eles poderão chegar em uma ordem diferente daquela em que foram enviados. Isso é algo que o transmissor provavelmente não esperava, e também pode chegar como uma surpresa (desagradável) ao receptor.

Esses tipos de diferenças podem ser ocultados com algum esforço. Por exemplo, um gateway juntando duas redes pode gerar pacotes separados para cada destino no lugar de melhor suporte de rede para multicasting. Um pacote grande poderia ser desmembrado, enviado em partes e depois montado de volta. Os receptores poderiam manter pacotes em buffer e entregá-los em ordem.

As redes também podem diferir em alguns aspectos que são mais difíceis de reconciliar. O exemplo mais claro é a qualidade de serviço. Se uma rede tem forte QoS e a outra oferece serviço de melhor esforço, será impossível fazer garantias de largura de banda e atraso para o tráfego em tempo real de ponto a ponto. De fato, eles provavelmente só podem ser feitos enquanto a rede de melhor esforço for operada em baixa utilização, ou quase nunca usada, o que provavelmente não é o objetivo da maioria dos ISPs. Os mecanismos de segurança são problemáticos, mas pelo menos a criptografia para confidencialidade e integridade de dados pode ser disposta sobre as redes que ainda não a incluíram. Finalmente, as diferenças na contabilidade podem levar a contas que não são bem recebidas quando o uso normal de repente se torna caro, como já descobriram os usuários de telefone móvel em *roaming* com planos de dados.

5.5.2 Como as redes podem ser conectadas

Existem duas escolhas básicas para conectar redes diferentes: podemos criar dispositivos que traduzam ou convertam pacotes de cada tipo de rede em pacotes para outra rede ou, como bons cientistas da computação, podemos tentar resolver o problema acrescentando uma camada de abstração, criando uma camada comum em cima das diferentes redes. De qualquer forma, os dispositivos são colocados nas fronteiras entre as redes.

Desde cedo, Cerf e Kahn (1974) argumentaram em favor de uma camada comum para ocultar as diferenças entre as redes existentes. Essa técnica tem sido tremendamente bem-sucedida, e a camada que eles propuseram por fim foi separada nos protocolos TCP e IP. Quase quatro décadas depois, o IP é o alicerce da Internet moderna. Por essa realização, Cerf e Kahn receberam o Turing Award de 2004, informalmente conhecido como o Prêmio Nobel da ciência da computação. O IP oferece um formato de pacote universal que todos os roteadores reconhecem e que pode ser passado por quase todas as redes. O IP estendeu seu alcance das redes de computadores para assumir a rede telefônica. Ele também funciona em redes de sensores e outros dispositivos pequenos que anteriormente eram considerados com muito poucos recursos para oferecer suporte.

Discutimos vários dispositivos que conectam as redes, incluindo repetidores, hubs, switches, bridges, roteadores e gateways. Os repetidores e hubs simplesmente movem

bits de um fio para outro. Eles são, em grande parte, dispositivos analógicos e não entendem coisa alguma sobre protocolos de camada mais alta. Bridges e switches operam na camada de enlace. Eles podem ser usados para criar redes, mas apenas com pouca intervenção do protocolo no processo, por exemplo, entre switches Ethernet de 10, 100 e 1.000 Mbps. Nosso foco nesta seção é a interconexão de dispositivos que operam na camada de rede, ou seja, os roteadores. Deixaremos os gateways, que são dispositivos de conexão de camada mais alta, para mais tarde.

Primeiro, exploraremos em alto nível como a interconexão com uma camada de rede comum pode ser usada para interconectar redes diferentes. Uma rede interligada composta de redes 802.11, MPSL e Ethernet aparece na Figura 5.34(a). Suponha que a máquina de origem na rede 802.11 queira enviar um pacote para a máquina de destino na rede Ethernet. Como essas tecnologias são diferentes, e elas são separadas ainda mais por outro tipo de rede (MPLS), é preciso haver algum processamento adicional nos limites entre as redes.

Como diferentes redes podem, em geral, ter diferentes formas de endereçamento, o pacote transporta um endereço da camada de rede que pode identificar qualquer host pelas três redes. O primeiro limite que o pacote alcança é quando ele faz a transição de uma rede 802.11 para uma rede MPLS. A 802.11 oferece um serviço não orientado a conexões, mas a rede MPLS oferece um serviço orientado a conexões. Isso significa que um circuito virtual precisa ser estabelecido para cruzar essa rede. Quando o pacote tiver atravessado o circuito virtual, ele alcançará a rede Ethernet. Nesse limite, o pacote pode ser muito grande para ser transportado, pois a 802.11 pode trabalhar com quadros maiores que a Ethernet. Para resolver esse problema, o pacote é dividido em fragmentos, e cada um deles é enviado separadamente. Quando os fragmentos alcançam o destino, eles são remontados. Então, o pacote completou sua jornada.

O processamento de protocolo para essa jornada aparece na Figura 5.34(b). A origem aceita dados da camada de transporte e gera um pacote com o cabeçalho comum da camada de rede, que, nesse exemplo, é o IP. O cabeçalho da rede contém o endereço de destino final, que é usado para determinar que o pacote deve ser enviado pelo primeiro roteador. Assim, o pacote é encapsulado em um quadro 802.11, cujo destino é o primeiro roteador, e transmitido. No roteador, o pacote é removido do campo de dados do quadro e o cabeçalho do quadro 802.11 é descartado. O roteador agora examina o endereço IP no pacote e pesquisa esse endereço em sua tabela de roteamento. Com base nesse endereço, ele decide enviar o pacote para o roteador seguinte. Para essa parte do caminho, um circuito virtual MPLS deve ser estabelecido para o segundo roteador e o pacote deve ser encapsulado com cabeçalhos MPLS que trafegam por esse circuito. No outro extremo, o cabeçalho MPLS é descartado e o endereço de rede é novamente consultado para encontrar o próximo hop da camada de rede. Esse é o próprio destino. Como o pacote é muito grande para ser enviado pela Ethernet, ele é dividido em duas partes. Cada uma dessas partes é colocada em um campo de dados de um quadro Ethernet e enviada para o endereço Ethernet do destino, onde o cabeçalho Ethernet tem cada um dos quadros removidos e o conteúdo é remontado. O pacote finalmente alcança seu destino.

Observe que existe uma diferença essencial entre o caso roteado e o caso comutado (bridge ou switch). Com um roteador, o pacote é extraído do quadro e o endereço da rede no pacote é usado para decidir para onde enviá-lo. Com um switch (ou bridge), o quadro inteiro é transportado com base em seu endereço MAC. Para comutar os pacotes, os switches não precisam entender o protocolo da camada de rede usado. Os roteadores sim.

Infelizmente, a interconexão de redes não é tão fácil quanto poderia parecer. Na verdade, quando as bridges fo-

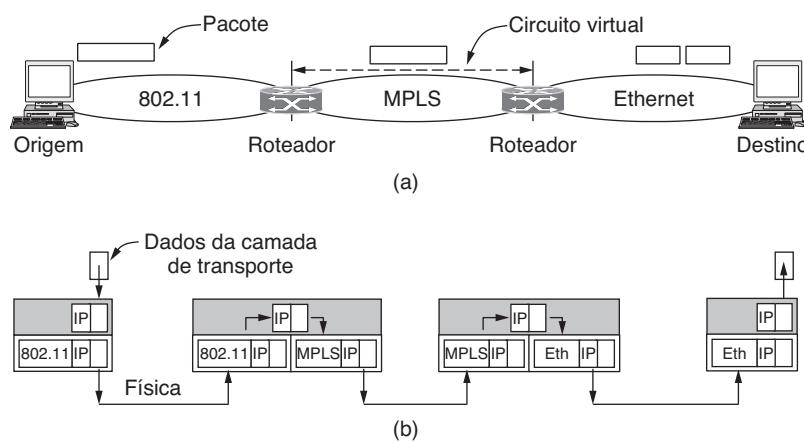



Figura 5.34 | (a) Um pacote cruzando diferentes redes. (b) Processamento de protocolo das camadas de rede e de enlace.

ram introduzidas, a intenção foi que elas unissem diferentes tipos de redes ou, pelo menos, diferentes tipos de LANs. Elas deveriam fazer isso traduzindo quadros de uma LAN para quadros de outra. Porém, isso não funcionou bem, pelo mesmo motivo de como a interligação de redes é difícil: as diferenças nos recursos das LANs, como diferentes tamanhos máximos de pacotes e LANs com e sem classes de prioridade, difíceis de mascarar. Hoje, as bridges são usadas predominantemente para conectar o mesmo tipo de rede na camada de enlace, e os roteadores conectam diferentes redes na camada de rede.

A interconexão de redes tem sido muito bem-sucedida na montagem de grandes redes, mas ela só funciona quando existe uma camada de rede comum. Na verdade, têm surgido muitos protocolos de rede com o tempo. É difícil fazer com que todos combinem com um único formato, quando as empresas percebem que é comercialmente proveitoso ter um formato próprio que elas controlam. Alguns exemplos além do IP, que agora é o protocolo de rede quase universal, foram IPX, SNA e AppleTalk. Nenhum desses protocolos continua sendo usado de forma generalizada, mas sempre há outros. Os exemplos mais relevantes agora provavelmente são o IPv4 e o IPv6. Embora ambos sejam versões do IP, eles não são compatíveis (ou então não teria sido necessário criar o IPv6).

Um roteador que pode lidar com vários protocolos de rede é chamado de **roteador multiprotocolos**. Ele precisa ou traduzir os protocolos ou permitir a conexão por um protocolo de camada mais alta. Nenhuma dessas técnicas é totalmente satisfatória. A conexão em uma camada mais alta, digamos, usando o TCP, requer que todas as redes o implementem (o que pode não ser o caso). Depois, ela limita o uso das redes em aplicações que usam TCP (o que não inclui muitas aplicações em tempo real).

A alternativa é traduzir pacotes entre as redes. Porém, a menos que os formatos de pacotes sejam parentes próximos com os mesmos campos de informação, tais conversões sempre serão incompletas e normalmente fadadas ao fracasso. Por exemplo, endereços IPv6 possuem 128 bits de extensão. Eles não caberão em um campo de endereço IPv4 de 32 bits, não importa quanto o roteador tente. Fazer com que IPv4 e IPv6 funcionem na mesma rede tem sido um grande obstáculo para a implementação do IPv6. (Para ser franco, o mesmo acontece para fazer com que os clientes entendam por que eles devem querer IPv6 em primeiro lugar.) Problemas maiores podem ser esperados quando se traduzem entre protocolos fundamentalmente diferentes, como protocolos de rede orientados e não orientados a conexões. Com essas dificuldades, a conversão raramente é experimentada. Comprovadamente, até mesmo o IP tem funcionado apenas o suficiente para servir como um tipo de denominador comum. Ele requer pouco das redes em que é executado, mas oferece apenas o melhor serviço possível como resultado.

5.5.3 Tunelamento

Lidar com a interligação de duas redes é extremamente difícil. Entretanto, existe um caso especial muito comum que proporciona bons resultados até mesmo para protocolos de rede diferentes. Isso acontece quando os hosts de origem e de destino estão no mesmo tipo de rede, mas há uma rede de outro tipo entre eles. Por exemplo, imagine um banco internacional com uma rede IPv6 em Paris, uma rede IPv6 em Londres e conectividade entre os escritórios por meio de Internet IPv4. Essa situação pode ser vista na Figura 5.35.

A solução para esse problema é uma técnica chamada **tunelamento** (tunneling). Para enviar um pacote IP a um host no escritório em Londres, um host em Paris constrói o pacote contendo um endereço IPv6 em Londres e o envia para o roteador multiprotocolo que conecta a rede IPv6 de Paris à Internet IPv4. Quando esse roteador recebe o pacote IPv6, ele o encapsula com um cabeçalho IPv4 endereçado ao lado IPv4 do roteador multiprotocolo que se conecta à rede IPv6 de Londres. Ou seja, o roteador coloca um pacote (IPv6) dentro de um pacote (IPv4). Quando esse pacote embrulhado chega, o roteador em Londres remove o pacote IPv6 original e o envia adiante para o host de destino.

O caminho pela Internet IPv4 pode ser visto como um grande túnel que se estende de um roteador multiprotocolo para o outro. O pacote IPv6 só trafega de uma extremidade do túnel para a outra, confortável em sua bela caixa. Ele não precisa, de forma alguma, se preocupar em lidar com o IPv4. O mesmo acontece com os hosts de Paris ou

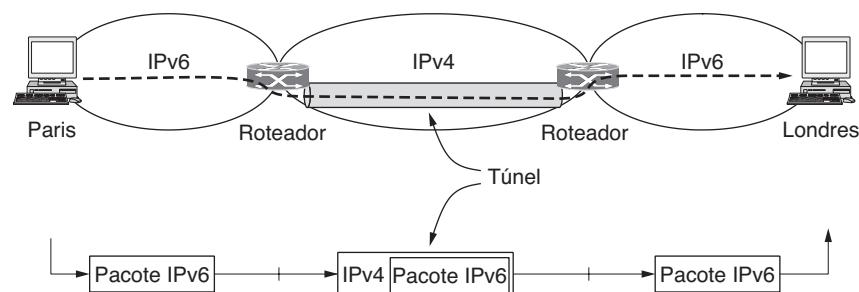



Figura 5.35 | Tunelamento de um pacote de Paris a Londres.

Londres. Somente os roteadores multiprotocolo precisam entender os pacotes IPv4 e IPv6. Com efeito, a viagem inteira de um roteador multiprotocolo para o outro é como um hop por um único enlace.

Uma analogia pode tornar o processo de tunelamento mais claro. Imagine uma pessoa dirigindo seu carro de Paris a Londres. Na França, o carro trafega em baixa velocidade, usando sua própria energia; no entanto, ao chegar ao Canal da Mancha, ele é colocado em um trem de alta velocidade e transportado para a Inglaterra pelo Eurotúnel (não é permitido o tráfego de automóveis nesse túnel). Na realidade, o carro está sendo transportado como uma carga, conforme mostra a Figura 5.36. Na outra extremidade, o carro passa a transitar nas estradas inglesas e continua a trafegar em velocidade baixa, com sua própria energia. Em uma rede externa, o tunelamento de pacotes funciona da mesma forma.

O tunelamento é bastante usado para conectar hosts e redes isoladas usando outras redes. A rede resultante é chamada **overlay**, pois efetivamente foi sobreposta em uma rede básica. A implantação de um protocolo de rede com um novo recurso é um motivo comum, como mostra o nosso exemplo 'IPv6 sobre IPv4'. A desvantagem do tunelamento é que nenhum dos hosts na rede, pertencentes ao túnel formado pelos hosts terminais, pode ser alcançado, pois os pacotes não podem escapar no meio do túnel. Contudo, essa limitação dos túneis é transformada em uma vantagem com as **VPNs** (**Virtual Private Networks**). Uma VPN é simplesmente um overlay usado para fornecer uma medida de segurança. Exploraremos as VPNs quando chegarmos ao Capítulo 8.

5.5.4 Roteamento entre redes

O roteamento por uma rede interligada é semelhante ao roteamento em uma única rede, mas há algumas outras complicações. Para começar, as redes podem usar internamente diferentes algoritmos de roteamento. Por exemplo, uma rede pode usar o roteamento de estado de enlace e outro roteamento por vetor de distância. Como os algoritmos de estado de enlace precisam conhecer a topologia, mas os algoritmos por vetor de distância não, essa diferença apenas não deixaria claro como encontrar os caminhos mais curtos pela rede interligada.

As redes usadas por diferentes operadores ocasionam problemas maiores. Primeiro, os operadores podem ter diferentes ideias sobre qual é um bom caminho pela rede. Um operador pode querer rotear com o menor atraso, enquanto outro pode querer a rota menos dispendiosa. Isso levará os operadores a usar diferentes quantidades para definir os custos do caminho mais curto (por exemplo, milissegundos de atraso *versus* custo monetário). Os pesos não serão comparáveis entre as redes, de modo que os caminhos mais curtos na rede interligada não serão bem definidos.

Pior ainda, um operador pode não querer que outro operador sequer conheça os detalhes dos caminhos em sua rede, talvez porque os pesos e os caminhos possam refletir informações confidenciais (como o custo monetário) que representam uma vantagem comercial competitiva.

Finalmente, a rede interligada pode ser muito maior do que qualquer uma das redes que a compreendem. Portanto, ela pode exigir algoritmos de roteamento que evoluem bem usando uma hierarquia, mesmo que nenhuma das redes individuais precise usar uma.

Todas essas considerações levam a um algoritmo de roteamento em dois níveis. Dentro de cada rede, um **protocolo de gateway interior** ou **intradomínio** é usado para o roteamento. ('Gateway' é um termo mais antigo para 'roteador'.) Ele poderia ser um protocolo de estado de enlace do tipo que já descrevemos. Entre as redes que compõem a rede interligada, é usado um **protocolo de gateway exterior** ou **interdomínio**. Todas as redes podem usar diferentes protocolos intradomínio, mas elas precisam usar o mesmo protocolo interdomínio. Na Internet, o protocolo de roteamento interdomínio é chamado **BGP** (**Border Gateway Protocol**). Vamos descrevê-lo na próxima seção.

Há mais um termo importante para apresentar. Como cada rede é operada independentemente de todas as outras, ela normalmente é chamada de **sistema autônomo**, ou **AS** (**Autonomous System**). Um bom modelo mental para um AS é uma rede de ISP. De fato, uma rede de ISP pode ser composta de mais de um AS, se for gerenciada, ou, se for adquirida, como múltiplas redes. Mas a diferença normalmente não é significativa.

Os dois níveis normalmente não são estritamente hierárquicos, pois caminhos muito abaixo do ideal podem resultar de uma grande rede internacional e uma peque-

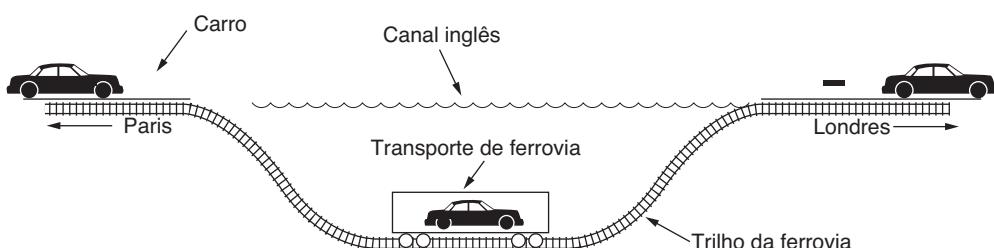



Figura 5.37 | Tunelamento de um carro da França até a Inglaterra.

na rede regional se ambos forem considerados uma única rede. Contudo, relativamente poucas informações sobre as rotas dentro das redes são expostas para encontrar as rotas pela rede interligada. Isso ajuda a resolver todas as complicações, pois melhora a escala e permite que os operadores selecionem livremente as rotas dentro de suas próprias redes usando um protocolo à sua escolha. Isso também não exige que os pesos sejam comparados entre as redes nem expõe informações confidenciais fora das redes.

Porém, dissemos pouco até aqui sobre como as rotas pelas redes da rede interligada são determinadas. Na Internet, um grande fator determinante são os arranjos comerciais entre os ISPs. Cada ISP pode cobrar ou receber dinheiro dos outros ISPs para transportar tráfego. Outro fator é que, se o roteamento da interligação de redes exigir a travessia de fronteiras internacionais, várias leis podem subitamente entrar em ação, como a lei de privacidade estrita da Suécia, sobre a exportação de dados pessoais sobre cidadãos suecos a partir da Suécia. Todos esses fatores não técnicos estão incluídos no conceito de uma **política de roteamento** que controla o modo como as redes autônomas selecionam as rotas que utilizam. Retornaremos às políticas de roteamento quando descrevermos o BGP.

5.5.5 Fragmentação de pacotes

Cada rede ou enlace impõe um tamanho máximo a seus pacotes. Entre as principais causas para essa limitação, temos:

1. Hardware (por exemplo, o tamanho de um quadro Ethernet).
2. Sistema operacional (por exemplo, todos os buffers têm 512 bytes).
3. Protocolos (por exemplo, o número de bits no campo de tamanho do pacote).
4. Compatibilidade com algum padrão (inter)nacional.
5. Desejo de reduzir de alguma forma as retransmissões provocadas por erros.
6. Desejo de evitar que um pacote ocupe o canal por muito tempo.

O resultado de todos esses fatores é que os projetistas de redes não têm liberdade para escolher o tamanho máximo de pacote que desejam. As cargas úteis máximas para algumas tecnologias comuns são de 1.500 bytes para Ethernet e 2.272 bytes para 802.11. O IP é mais generoso, e permite pacotes com até 65.515 bytes.

Os hosts normalmente preferem transmitir pacotes grandes, pois isso reduz os overheads de pacote, como largura de banda desperdiçada em bytes de cabeçalho. Um problema óbvio de rede interligada aparece quando um pacote grande deseja atravessar uma rede cujo tamanho de pacote máximo é muito pequeno. Esse incômodo tem sido um problema persistente, e as soluções para ele têm evoluído com a grande experiência obtida na Internet.

Uma solução é garantir que o problema não ocorra em primeiro lugar. Porém, é mais fácil falar do que fazer. Uma origem normalmente não conhece o caminho que um pacote tomará na rede até um destino, de modo que certamente não sabe o tamanho que os pacotes devem ter para que cheguem lá. Esse tamanho de pacote é chamado **unidade máxima de transmissão do caminho**, ou **Path MTU** (**Path Maximum Transmission Unit**). Mesmo que a origem não saiba a MTU do caminho, os pacotes são roteados independentemente em uma rede não orientada a conexões, como a Internet. Esse roteamento significa que os caminhos de repente podem mudar, o que pode mudar inesperadamente a MTU do caminho.

A solução alternativa para o problema é permitir que os roteadores quebrem os pacotes em **fragmentos**, enviando cada um deles como um pacote separado da camada de rede. Porém, como todo pai de filho pequeno sabe, converter um objeto grande em pequenos fragmentos é muito mais fácil do que o processo inverso. (Os físicos até mesmo deram um nome a esse efeito: a segunda lei da termodinâmica.) As redes de comutação de pacotes também têm problemas para montar os fragmentos de volta.

Existem duas estratégias opostas para recombinar os fragmentos de volta ao pacote original. A primeira estratégia é tornar a fragmentação causada por uma rede de 'pacotes pequenos' transparente a quaisquer redes subsequentes através das quais o pacote deve passar em seu caminho até o destino final. Essa opção é mostrada na Figura 5.37(a). Nessa técnica, quando um pacote de tamanho superior chega a *G1*, o roteador o desmembra em fragmentos. Cada um deles é endereçado para o mesmo roteador de saída, *G2*, onde as partes são recombinadas. Desse modo, a passagem pela rede de pacotes pequenos se torna transparente. As redes subsequentes nem sequer sabem que houve fragmentação.

A fragmentação transparente é simples, mas apresenta alguns problemas. Por um lado, o roteador de saída deve saber quando recebeu todas as partes; portanto, é necessário incluir um campo de contagem ou um bit de 'fim de pacote' em cada um deles. Além disso, como todos os pacotes têm de sair pelo mesmo roteador para que possam ser reconstruídos, as rotas são restritas. Como não é permitido que alguns fragmentos sigam uma rota até o destino final e que outros fragmentos percorram uma rota distinta, há uma perda considerável em termos de desempenho. Mais significativa é a quantidade de trabalho que o roteador pode ter de fazer. Ele pode precisar manter os fragmentos em buffer assim que chegam e decidir quando descartá-los, se nem todos os fragmentos chegarem. Parte desse trabalho também pode ser desperdiçada, pois o pacote pode passar por uma série de redes de pequenos pacotes, tendo de ser repetidamente fragmentado e reconstruído.

A outra estratégia de fragmentação é evitar recombinar os fragmentos em qualquer roteador intermediário. Quando um pacote é fragmentado, cada fragmento é tra-

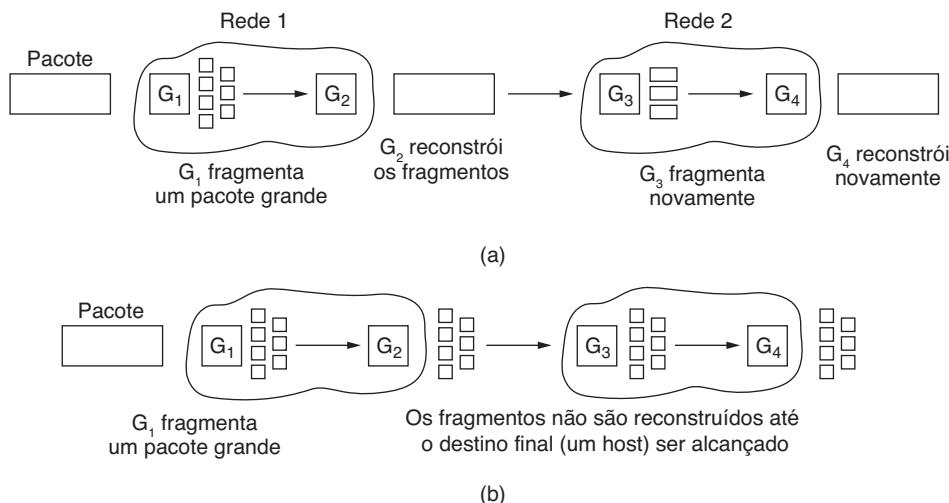



Figura 5.37 | (a) Fragmentação transparente. (b) Fragmentação não transparente.

tado como se fosse o pacote original. Os roteadores passam os fragmentos, como mostra a Figura 5.37(b), e a reconstrução ocorre apenas no host de destino.

A principal vantagem da fragmentação não transparente é que ela exige que os roteadores realizem menos trabalho. O IP funciona assim. Um projeto completo requer que os fragmentos sejam numerados de modo que o fluxo de dados original possa ser reconstruído. O projeto usado pelo IP é dar a cada fragmento um número de pacote (transportado em todos os pacotes), um deslocamento de byte absoluto dentro do pacote e uma flag indicando se esse é o final do pacote. Um exemplo aparece na Figura 5.38. Embora simples, esse projeto tem algumas propriedades atraentes. Os fragmentos podem ser colocados em um buffer no destino, no local certo para a reconstrução, mesmo que eles cheguem fora de ordem. Os fragmentos também podem ser subdivdidos se passarem por uma rede com uma MTU ainda menor. Isso pode ser visto na Figura 5.38(c). As retransmissões do pacote (se todos os fragmentos não fossem recebidos) podem ser fragmentadas em diferentes partes. Finalmente, os fragmentos podem ter um tamanho qualquer, até um de único byte mais o cabeçalho do pacote. Em todos os casos, o destino simplesmente usa o número do pacote e o deslocamento de fragmento para colocar os dados na posição correta, e a flag de final de pacote para determinar quando ele tem o pacote completo.

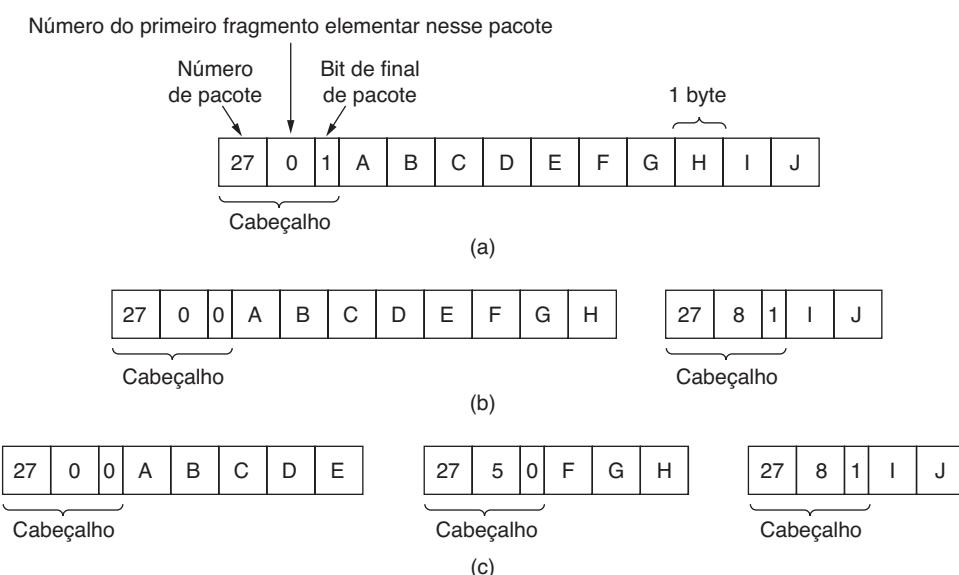



Figura 5.38 | Fragmentação quando o tamanho de dados elementares é de 1 byte. (a) Pacote original, contendo 10 bytes de dados. (b) Fragmentos após passarem por uma rede com tamanho máximo de pacote de 8 bytes de carga útil mais cabeçalho. (c) Fragmentos após passarem por um gateway de tamanho 5.

Infelizmente, esse projeto ainda tem problemas. O overhead pode ser maior do que com a fragmentação transparente, pois os cabeçalhos de fragmento agora são transportados por alguns enlaces em que eles podem não ser necessários. Mas o problema real é a existência de fragmentos em primeiro lugar. Kent e Mogul (1987) argumentaram que essa fragmentação prejudica o desempenho porque, assim como os overheads do cabeçalho, um pacote inteiro é perdido se qualquer um de seus fragmentos for perdido, e também porque a fragmentação é mais um peso para os hosts do que o que foi observado originalmente.

Isso nos leva de volta à solução original de remover a fragmentação na rede, a estratégia usada na Internet moderna. O processo é chamado **descoberta da MTU do caminho** (Mogul e Deering, 1990). Ele funciona da seguinte forma: cada pacote IP é enviado com seus bits de cabeçalho definidos para indicar que nenhuma fragmentação poderá ser realizada. Se um roteador recebe um pacote muito grande, ele gera um pacote de erro, retorna-o para a origem e remove o pacote. Isso é mostrado na Figura 5.39. Quando a origem recebe o pacote de erro, ela usa a informação no interior para refragmentar o pacote em partes pequenas o suficiente para o roteador tratar. Se um roteador mais adiante no caminho tiver uma MTU ainda menor, o processo é repetido.

A vantagem da descoberta da MTU do caminho é que a origem agora sabe que tamanho de pacote enviar. Se as rotas e MTU do caminho mudarem, novos pacotes de erro serão disparados e a origem se adaptará ao novo caminho. Porém, a fragmentação ainda é necessária entre a origem e o destino, a menos que as camadas mais altas descubram a MTU do caminho e passem a quantidade certa de dados ao IP. TCP e IP normalmente são implementados juntos (como 'TCP/IP') para poder passar esse tipo de informação. Mesmo que isso não seja feito para outros protocolos, a fragmentação ainda terá sido passada da rede para os hosts.

A desvantagem da descoberta da MTU do caminho é que pode haver outros atrasos de saída simplesmente para enviar um pacote. Mais de um atraso de ida e volta podem ser necessários para sondar o caminho e encontrar a MTU antes que qualquer dado seja entregue ao destino. Isso levanta a questão da existência de projetos melhores. A resposta provavelmente é 'Sim'. Considere o projeto em que cada roteador simplesmente trunca os pacotes que excedem sua MTU. Isso garantiria que o destino descobrisse a MTU o mais rápido possível (a partir da quantidade de dados que foram descartados) e recebesse alguns dos dados.

5.6 A camada de rede da Internet

Agora é hora de discutirmos a camada de rede da Internet detalhadamente. Porém, antes de entrarmos nesses detalhes, vale a pena dar uma olhada nos princípios que controlaram seu projeto no passado e a tornaram o sucesso que ela é hoje. Com frequência, as pessoas parecem ter se esquecido deles. Esses princípios são enumerados e discutidos na RFC 1958, que merece ser lida (e isso deve ser obrigatório para todos os projetistas de protocolo — com uma prova final). Essa RFC se baseia bastante nas ideias estabelecidas por Clark (1988) e Saltzer et al. (1984). Agora, vamos resumir o que consideramos ser os dez maiores princípios (do mais ao menos importante).

1. **Certifique-se de que funciona**. Não conclua o projeto ou o padrão até que vários protótipos tenham conseguido se comunicar com sucesso uns com os outros. Um número muito grande de projetistas escreve no início um padrão de mil páginas, obtém sua aprovação e depois descobre que ele tem falhas profundas e não funciona. Então, esses projetistas escrevem a versão 1.1 do padrão. Esse não é o melhor caminho.
2. **Mantenha a simplicidade**. Quando estiver em dúvida, use a solução mais simples. William de Occam enunciou este princípio (a navalha de Occam) no século XIV. Em termos modernos: os recursos entram em conflito. Se um recurso não for absolutamente essencial, deixe-o de fora, em especial se o mesmo efeito puder ser obtido pela combinação de outros recursos.
3. **Faça escolhas claras**. Se houver várias maneiras de executar a mesma ação, escolha apenas uma. Ter duas ou mais opções para realizar a mesma ação é procurar problemas. Com frequência, os padrões têm diferentes opções, modos ou parâmetros, porque várias partes poderosas insistem em afirmar que sua alternativa é a melhor. Os projetistas devem resistir com firmeza a essa tendência. Basta dizer não.
4. **Explore a modularidade**. Esse princípio leva diretamente à ideia de pilhas de protocolos, em que cada uma das camadas é independente de todas as outras. Desse modo, se as circunstâncias exigirem mudanças em um módulo ou em uma camada, os outros itens não serão afetados.

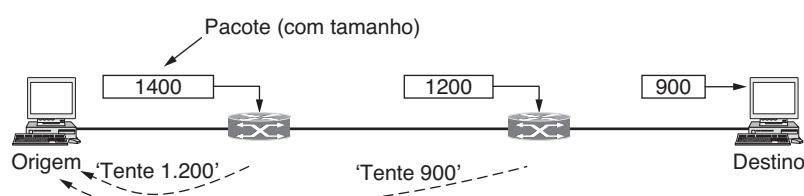



Figura 5.39 | Descoberta da MTU do caminho.

5. **Espere heterogeneidade**. Diferentes tipos de hardware, instalações de transmissão e aplicações ocorrerão em qualquer rede de grande porte. Para lidar com isso, o projeto de rede deve ser simples, geral e flexível.
6. **Evite opções e parâmetros estáticos**. Se os parâmetros forem inevitáveis (por exemplo, tamanho máximo de pacote), é melhor fazer o transmissor e o receptor negociar um valor em vez de definir opções fixas.
7. **Procure um bom projeto**; **ele não precisa ser perfeito**. Frequentemente, os projetistas têm um bom projeto, mas não conseguem lidar com algum caso especial complicado. Em vez de alterar o projeto, eles devem dar continuidade ao bom projeto e entregar o fardo de trabalhar com ele às pessoas que fizeram as exigências complexas.
8. **Seja rígido ao enviar e tolerante ao receber**. Em outras palavras, só envie pacotes que obedeçam rigorosamente aos padrões, mas espere receber pacotes que talvez não sejam plenamente compatíveis e procure lidar com eles.
9. **Pense na escalabilidade**. Se o sistema tiver de manipular milhões de hosts e bilhões de usuários de forma efetiva, nenhum banco de dados centralizado de qualquer tipo será tolerável, e a carga deverá ser espalhada da maneira mais uniforme possível pelos recursos disponíveis.
10. **Considere desempenho e custo**. Se uma rede tiver fraco desempenho ou custos exagerados, ninguém a usará.

Agora vamos deixar de lado os princípios gerais e iniciar o exame dos detalhes da camada de rede da Internet. Nessa camada, a Internet pode ser vista como um conjunto de sub-redes ou **sistemas autônomos**, ou **AS** (**Autonomous Systems**), conectados entre si. Não existe uma estrutura real, mas diversos backbones principais, construídos a partir de linhas de grande largura de banda e roteadores rápidos. O maior desses backbones, ao qual todos os outros se conectam para alcançar o restante da Internet, é chamado **rede de nível 1**. Conectados aos backbones estão os ISPs (Internet Service Providers), que oferecem acesso à Internet para casas e empresas, centros de dados e instalações repletas de máquinas servidoras, e redes regionais (de nível intermediário). Os centros de dados oferecem grande parte do conteúdo enviado pela Internet. Conectados às redes regionais estão mais ISPs, LANs em muitas universidades e empresas, além de outras redes na borda. Um esquema dessa organização semi-hierárquica é mostrado na Figura 5.40.

O elemento que mantém a Internet unida é o protocolo da camada de rede, o **IP** (**Internet Protocol**). Ao contrário da maioria dos protocolos da camada de rede mais antigos, o IP foi projetado desde o início tendo como objetivo a interligação de redes. Uma boa maneira de pensar na camada de rede é esta: sua tarefa é fornecer a melhor forma possível (ou seja, sem garantias) de transportar pacotes da origem para o destino, independentemente de essas máquinas estarem na mesma rede ou de haver outras redes entre elas.

Na Internet, a comunicação funciona da forma descrita a seguir. A camada de transporte recebe os fluxos de dados e os divide para que possam ser enviados como pacotes IP. Teoricamente, os pacotes podem ter até 64 KB; no entanto,

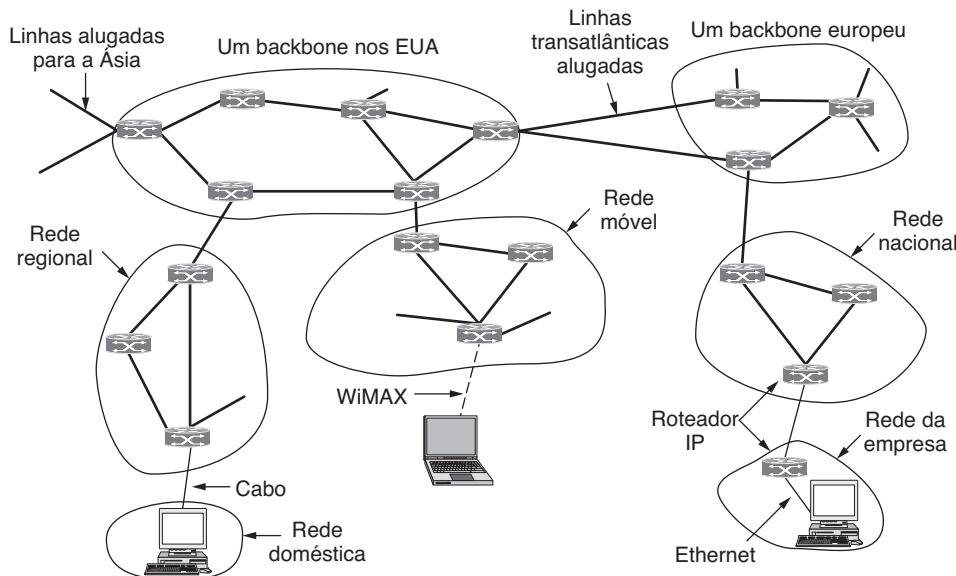



Figura 5.40 | A Internet é uma coleção interconectada de muitas redes.

na prática, eles geralmente têm no máximo 1.500 bytes (e, portanto, cabem em um único quadro Ethernet). Os roteadores IP encaminham cada pacote pela Internet, por um caminho de um roteador para o seguinte, até que o destino seja alcançado. No destino, a camada de rede entrega os dados à camada de transporte, que os oferece ao processo receptor. Quando todos os fragmentos finalmente chegam à máquina de destino, eles são remontados pela camada de rede no datagrama original. Esse datagrama é, então, entregue à camada de transporte.

No exemplo da Figura 5.40, um pacote originando-se em um host na rede doméstica precisa atravessar quatro redes e um grande número de roteadores IP antes que consiga chegar à rede da empresa na qual o host de destino está localizado. Isso não é raro na prática, e existem muitos caminhos maiores. Também há muita conectividade redundante na Internet, com backbones e ISPs conectando-se uns aos outros em diversos locais. Isso significa que existem muitos caminhos possíveis entre dois hosts. A tarefa dos protocolos de roteamento IP é decidir quais caminhos serão usados.

5.6.1 O protocolo IP versão 4 (IPv4)

Um local apropriado para iniciar nosso estudo da camada de rede da Internet é o formato dos próprios datagramas IP. Um datagrama IPv4 consiste em uma parte de cabeçalho e uma parte de dados. O cabeçalho tem uma parte fixa de 20 bytes e uma parte opcional de tamanho variável. O formato do cabeçalho é mostrado na Figura 5.41. Os bits são transmitidos da esquerda para a direita e de cima para baixo, com o bit de mais alta ordem do campo *Versão* aparecendo primeiro. (Essa é uma ordem de byte de rede 'big-endian'. Em máquinas 'little-endian', como os computadores x86 da Intel, uma conversão de software é exibida na transmissão e na recepção.) Fazendo um retrospecto, 'little-endian' teria sido uma escolha melhor, mas, quando o IP foi projetado, ninguém saiba que ele viria a dominar a computação.

O campo *Versão* controla a versão do protocolo à qual o datagrama pertence. A versão 4 domina a Internet hoje, e foi aí que começamos nossa discussão. Incluindo a versão no início de cada datagrama, é possível ter uma transição entre as versões por um longo período. Na verdade, o IPv6, a próxima versão do IP, foi definido há mais de uma década, embora ainda esteja só começando a ser implementado. Vamos descrevê-lo mais adiante nesta seção. Seu uso por fim será forçado quando cada uma das quase 2^{31} pessoas na China tiverem um PC desktop, um laptop e um telefone IP. A propósito da numeração, o IPv5 era um protocolo de fluxo em tempo real experimental, e nunca foi amplamente utilizado.

Como o tamanho do cabeçalho não é constante, existe um campo no cabeçalho, *IHL*, que informa seu tamanho em palavras de 32 bits. O valor mínimo é 5, quando não há nenhuma opção presente. O valor máximo desse campo de 4 bits é 15, que limita o cabeçalho a 60 bytes, e, portanto, o campo *Opções* a 40 bytes. Para algumas opções, como aquela que registra a rota que um pacote tomou, 40 bytes é muito pouco, tornando essas opções inúteis.

O campo *Serviços diferenciados* é um dos poucos campos que mudaram (ligeiramente) seu significado com o passar dos anos. Originalmente, ele se chamava *Tipo de serviço*. Ele foi e ainda é destinado a distinguir entre diferentes classes de serviços. São possíveis várias combinações de confiabilidade e velocidade. Em se tratando de voz digitalizada, a entrega rápida vence a entrega segura. Para a transferência de arquivos, uma transmissão sem erros é mais importante do que uma transmissão rápida. O campo *Tipo de serviço* fornecia 3 bits para prioridade de sinal e 3 bits para sinalizar se um host se importava mais com atraso, throughput ou confiabilidade. Porém, ninguém realmente sabia o que fazer com esses bits nos roteadores, de modo que ficaram sem uso por muitos anos. Quando os serviços diferenciados foram projetados, a IETF jogou a toalha e reutilizou esse campo. Agora, os seis bits superiores são usados para marcar o pacote com sua classe de serviço; descrevemos os serviços expressos e garantidos anteriormente neste ca-

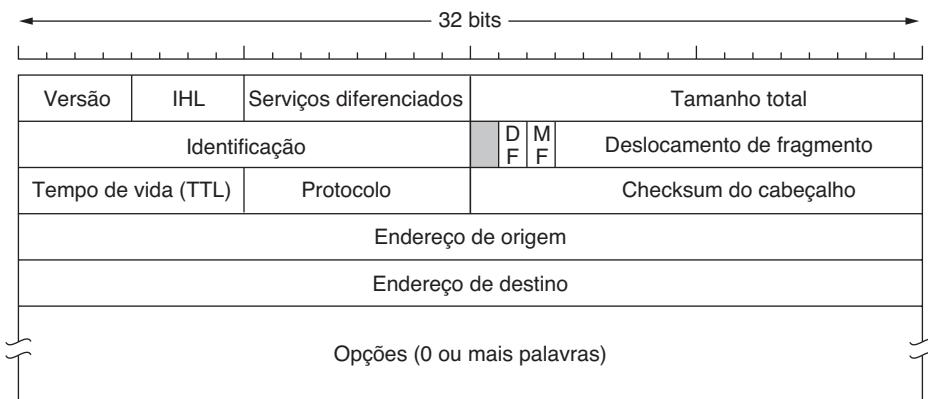



Figura 5.41 | O cabeçalho IPv4 (Internet Protocol).

pítulo. Os 2 bits inferiores são usados para transportar informações explícitas de notificação de congestionamento, como se o pacote a tivesse experimentado; descrevemos essa notificação explícita como parte do controle de congestionamento anteriormente neste capítulo.

O campo *Tamanho total* inclui tudo o que há no datagrama — cabeçalho e dados. O tamanho máximo é de 65.535 bytes. Atualmente, esse limite superior é tolerável, mas, com as futuras redes de gigabits, serão necessários datagramas maiores.

O campo *Identificação* é necessário para permitir que o host de destino determine a qual datagrama pertence um fragmento recém-chegado. Todos os fragmentos de um datagrama contêm o mesmo valor de *Identificação*.

Em seguida, há um bit não utilizado, o que é surpreendente, pois o espaço disponível no cabeçalho IP é extremamente escasso. Como uma piada de primeiro de abril, Bellovin (2003) propôs o uso desse bit para detectar tráfego malicioso. Isso simplificaria bastante a segurança, pois os pacotes com o bit 'malicioso' marcado seriam conhecidos por ter sido enviados por invasores e poderiam simplesmente ser descartados. Infelizmente, a segurança na rede não é tão simples assim.

Depois vêm dois campos de 1 bit relacionados à fragmentação. *DF* significa *Don't Fragment* (Não Fragmentar). Trata-se de uma ordem para os roteadores não fragmentarem o datagrama. Originalmente, a intenção era dar suporte a hosts incapazes de juntar as partes novamente. Agora ele é usado como parte do processo de descobrir a MTU do caminho, que é o maior pacote que pode atravessar um caminho sem ser fragmentado. Marcando o datagrama com o bit *DF*, o transmissor sabe que ele chegará a uma só parte, ou uma mensagem de erro será retornada ao transmissor.

MF significa Mais Fragmentos. Todos os fragmentos, exceto o último, têm esse conjunto de bits, necessário para saber quando chegaram todos os fragmentos de um datagrama.

O campo *Deslocamento de fragmento* informa a que ponto do datagrama atual o fragmento pertence. Todos os fragmentos de um datagrama, com exceção do último, devem ser múltiplos de 8 bytes, a unidade elementar de fragmento. Como são fornecidos 13 bits, existem no máximo 8.192 fragmentos por datagrama, resultando em um tamanho máximo de pacote igual ao limite do campo *Tamanho total*. Trabalhando juntos, os campos *Identificação*, *MF* e *Deslocamento de fragmento* são usados para implementar a fragmentação descrita na Seção 5.5.5.

O campo *TTL* (*Time to Live*) é um contador usado para limitar a vida útil dos pacotes. Esse campo originalmente deveria contar o tempo em segundos, permitindo uma vida útil máxima de 255s. Esse contador deve ser decrementado a cada hop e supõe-se que ele seja decrementado diversas vezes quando estiver enfileirado durante um longo tempo em um roteador. Na prática, ele simplesmente conta os hops. Quando o contador chega a zero, o pacote é descartado e um pacote de advertência é enviado ao host de origem. Esse recurso evita que os datagramas fiquem vagando indefinidamente, algo que aconteceria se as tabelas de roteamento fossem danificadas.

Quando tiver montado um pacote completo, a camada de rede precisará saber o que fazer com ele. O campo *Protocolo* informa a que processo de transporte o datagrama deve ser entregue. O TCP é uma possibilidade, mas também há o UDP e alguns outros. A numeração dos protocolos se aplica a toda a Internet. Os protocolos e outros números atribuídos foram listados inicialmente na RFC 1700, mas hoje eles estão contidos em um banco de dados on-line localizado em www.iana.org.

Como o cabeçalho transporta informações vitais, como endereços, ele contém seu próprio checksum por proteção, o *Checksum do cabeçalho*. O algoritmo tem como objetivo somar todas as meias palavras de 16 bits do cabeçalho à medida que elas chegam, utilizando a aritmética de complemento de um e, depois, calculando o complemento de um do resultado. Para os propósitos desse algoritmo, supomos que esse campo seja zero no momento da chegada. Esse checksum é útil para detectar erros enquanto o pacote atravessa a rede. Observe que ele deve ser recontado a cada hop, porque pelo menos um campo sempre se altera (o campo *TTL*), mas existem artifícios que podem ser usados para acelerar o cálculo.

Os campos *Endereço de origem* e *Endereço de destino* indicam o endereço IP das interfaces de rede de origem e destino. Discutiremos os endereços da Internet na próxima seção.

O campo *Opções* foi projetado para permitir que versões posteriores do protocolo incluam informações inexistentes no projeto original, possibilitando a experimentação de novas ideias e evitando a alocação de bits de cabeçalho para informações raramente necessárias. Existem opções de tamanhos variáveis. Cada uma começa com um código de 1 byte identificando a opção. Algumas opções são seguidas por um campo de tamanho de opção de 1 byte e, depois, por 1 ou mais bytes de dados. O campo *Opções* é preenchido até alcançar um múltiplo de 4 bytes. Originalmente, havia cinco opções definidas, como mostra a Tabela 5.5.

A opção *Security* mostra o nível de segurança da informação. Teoricamente, um roteador militar poderia usar esse campo para especificar que não se devem seguir rotas que passam por certos países que os militares consideram 'mal-comportados'. Na prática, todos os roteadores a ignoram, pois sua única função prática é ajudar os espiões a descobrir mais facilmente onde estão as melhores informações.

A opção *Strict source routing* fornece o caminho completo da origem ao destino como uma sequência de endereços IP. O datagrama é obrigado a seguir exatamente essa rota. Essa opção é mais útil principalmente para os gerentes de sistemas enviarem pacotes de emergência quando as tabelas de roteamento estão danificadas ou para fazer medições de sincronização.

Opção	Descrição
Security	Especifica o nível de segurança do datagrama
Strict source routing	Mostra o caminho completo a ser seguido
Loose source routing	Apresenta uma lista de roteadores que não devem ser esquecidos
Record route	Faz com que cada roteador anexe seu endereço IP
Timestamp	Faz com que cada roteador anexe seu endereço e seu registro de tempo de tempo (flag)

Tabela 5.5 | Algumas opções do IP.

A opção *Loose source routing* exige que o pacote percorra uma lista de roteadores específicos, na ordem especificada, mas permite que ele passe por outros roteadores durante o percurso. Normalmente, essa opção forneceria um pequeno número de roteadores, a fim de forçar determinado caminho. Por exemplo, se for necessário forçar um pacote a ir de Londres até Sydney pelo oeste e não pelo leste, essa opção poderia especificar roteadores em Nova York, Los Angeles e Honolulu. A opção é útil principalmente quando considerações políticas ou econômicas exigem a passagem por certos países ou que eles sejam evitados.

A opção *Record route* informa aos roteadores ao longo do caminho que eles devem anexar seu endereço IP ao campo *Opções*. Isso permite que administradores de sistemas depurem algoritmos de roteamento ("Por que os pacotes de Houston para Dallas estão passando primeiro por Tóquio?"). Quando a ARPANET foi criada, nenhum pacote passava por mais de nove roteadores; por isso, 40 bytes de opção eram mais do que suficientes. Como mencionado anteriormente, agora esse espaço é muito pequeno.

Por fim, a opção *Timestamp* é semelhante à opção *Record route*, exceto pelo fato de, além de registrar seu endereço IP de 32 bits, cada roteador também mantém um registro de tempo de 32 bits. Essa opção também é mais útil para medição da rede.

Hoje, as opções do IP perderam popularidade. Muitos roteadores as ignoram ou não as processam de modo eficiente, deixando-as de lado como um caso incomum. Ou seja, elas são aceitas apenas parcialmente e raramente são usadas.

5.6.2 Endereços IP

Um recurso que define o IPv4 são seus endereços de 32 bits. Cada host e roteador na Internet tem um endereço IP que pode ser usado nos campos *Endereço de origem* e *Endereço de destino* dos pacotes IP. É importante observar que um endereço IP não se refere realmente a um host. Na verdade, ele se refere a uma interface de rede; assim, se um host estiver em duas redes, ele precisará ter dois endereços IP. Porém, na prática, a maioria dos hosts está em uma única rede e, portanto, só tem um endereço IP. Ao contrário, os roteadores têm várias interfaces e, portanto, vários endereços IP.

Prefixos

Os endereços IP são hierárquicos, diferentes dos endereços Ethernet. Cada endereço de 32 bits é composto de uma parte de rede de tamanho variável nos bits superiores e uma parte de host nos bits inferiores. A parte de rede tem o mesmo valor para todos os hosts em uma única rede, como uma LAN Ethernet. Isso significa que uma rede corresponde a um bloco contíguo de espaço de endereços IP. Esse bloco é chamado de **prefixo**.

Os endereços IP são escritos em **notação decimal com ponto**. Nesse formato, cada um dos 4 bytes é escrito em decimal, de 0 a 255. Por exemplo, o endereço hexadecimal de 32 bits 80D00297 é escrito como 128.208.2.151. Os prefixos são escritos dando o menor endereço IP no bloco de endereços. O tamanho do prefixo é determinado pelo número de bits na parte de rede; os bits restantes fazem parte do campo de host e podem variar. Isso significa que o tamanho do endereço deve ser uma potência de dois. Por convenção, ele é escrito após o prefixo com uma barra seguida pelo tamanho em bits da parte da rede. Em nosso exemplo, se o prefixo tiver 2^8 endereços e, portanto, deixar 24 bits para a parte de rede, ele é escrito como 128.208.0.0/24.

Como o tamanho do prefixo não pode ser deduzido apenas pelo endereço IP, os protocolos de roteamento devem transportar os prefixos aos roteadores. Às vezes, os prefixos são simplesmente descritos por seu tamanho, como em um '/16' que é pronunciado como 'barra 16'. O tamanho do prefixo corresponde a uma máscara binária de 1s na parte destinada à rede. Quando escrita dessa forma, ela é chamada **máscara de sub-rede**. Ela pode ser submetida a um AND com o endereço IP a fim de extrair apenas a parte da rede do endereço IP. Para nosso exemplo, a máscara de sub-rede é 255.255.255.0. A Figura 5.42 mostra um prefixo e uma máscara de sub-rede.

Endereços hierárquicos têm vantagens e desvantagens significativas. A principal vantagem dos prefixos é que os roteadores podem encaminhar pacotes com base apenas na parte de rede do endereço, desde que cada uma das redes tenha um bloco de endereços exclusivos. A parte do host não importa para os roteadores, pois todos os hosts na mes-

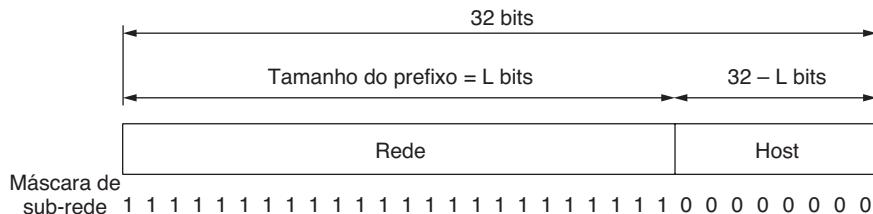



Figura 5.42 | Um prefixo IP e uma máscara de sub-rede.

ma rede estarão na mesma direção. Somente quando os pacotes chegam à rede para a qual são destinados é que eles são encaminhados para o host correto. Isso torna as tabelas de roteamento muito menores do que elas seriam de outra forma. Considere que o número de hosts na Internet está se aproximando de um bilhão. Essa seria uma tabela muito grande para cada roteador manter. Porém, usando uma hierarquia, os roteadores precisam manter rotas somente para cerca de 300 mil prefixos.

Embora o uso de uma hierarquia permita a expansão do roteamento na Internet, ele tem duas desvantagens. Primeiro, o endereço IP de um host depende do local onde ele está na rede. Um endereço Ethernet pode ser usado em qualquer lugar do mundo, mas cada endereço IP pertence a uma rede específica, e os roteadores só poderão entregar pacotes destinados a esse endereço na rede. Projetos como o IP móvel são necessários para dar suporte a hosts que se movem entre as redes, mas que querem manter os mesmos endereços IP.

A segunda desvantagem é que a hierarquia desperdiça endereços, a menos que seja cuidadosamente gerenciada. Se os endereços são atribuídos a redes em blocos (muito) grandes, haverá (muitos) endereços que são alocados, mas não usados. Essa alocação não importaria muito se houvesse muitos endereços para utilizar. Porém, há mais de duas décadas foi observado que o tremendo crescimento da Internet estava esgotando rapidamente o espaço de endereços livres. O IPv6 é a solução para essa escassez, mas, até que ele seja largamente implantado, haverá muita pressão para alocar endereços IP de modo que sejam usados de maneira mais eficiente.

Sub-redes

Os números de rede são controlados por uma corporação não lucrativa chamada **ICANN** (**Internet Corporation for Assigned Names and Numbers**), para evitar conflitos. Por sua vez, a ICANN delegou partes do espaço de endereços a diversas autoridades regionais, que distribuíram endereços IP aos ISPs e a outras empresas. Esse é o processo pelo qual uma empresa aloca um bloco de endereços IP.

Porém, esse processo é apenas o começo da história, pois a atribuição de endereços IP é contínua, à medida que as empresas crescem. Dissemos que o roteamento por prefixo exige que todos os hosts de uma rede tenham o mesmo número de rede. Essa propriedade poderá causar problemas à medida que as redes crescem. Por exemplo, imagine uma universidade que começou com nosso exemplo de prefixo /16 para uso pelo departamento de ciência da computação, para os computadores em sua Ethernet. Um ano mais tarde, o departamento de engenharia elétrica quis entrar na Internet, e o mesmo aconteceu com o departamento de artes. Que endereços IP esses departamentos deverão usar? Para obter outros blocos, seria preciso sair da universidade, e isso pode ser caro e inconveniente. Além do mais, o /16 já alocado tem endereços suficientes para mais de 60.000 hosts. A intenção poderia ser permitir um crescimento significativo, mas, até que isso aconteça, é um desperdício alocar mais blocos de endereços IP à mesma universidade. É necessária uma organização diferente.

A solução para esses problemas é permitir que uma rede seja dividida em diversas partes para uso interno, mas externamente continue a funcionar como uma única rede. Isso é **subdivisão de rede**, e as redes (como as LANs Ethernet) que resultam da divisão de uma rede maior são chamadas **sub-redes**. Como dissemos no Capítulo 1, você precisa estar ciente de que esse novo uso do termo entra em conflito com o uso mais antigo de 'sub-rede', que significa o conjunto de todos os roteadores e enlaces de comunicação em uma rede.

A Figura 5.43 mostra como as sub-redes podem ajudar em nosso exemplo. O único /16 foi dividido em partes. Essa divisão não precisa ser uniforme, mas cada parte precisa estar alinhada de modo que os bits possam ser usados na parte destinada ao host. Nesse caso, metade do bloco (/17) é alocada ao departamento de ciência da computação, um quarto é alocado ao departamento de engenharia elétrica (/18) e um oitavo (/19), ao departamento de artes. O oitavo restante não é alocado. Um modo diferente de ver como o bloco foi dividido é examinar os prefixos resultantes quando escritos em notação binária:

Ciência da computação:	10000000	11010000	1|xxxxxxx	xxxxxxxx
Engenharia elétrica:	10000000	11010000	00|xxxxxx	xxxxxxxx
Artes:	10000000	11010000	011|xxxxx	xxxxxxxx

Aqui, a barra vertical (|) mostra o limite entre a parte da sub-rede e a parte do host.

Quando um pacote entra no roteador principal, como ele sabe a qual sub-rede deve ir? É aí que entram os detalhes de nossos prefixos. Uma maneira seria cada roteador ter uma tabela com 65.536 entradas, informando qual interface de saída usar para cada host no campus. Mas isso prejudicaria o benefício principal da expansão que obtemos com o uso de uma hierarquia. Em vez disso, os roteadores simplesmente precisam conhecer as máscaras de sub-rede para as redes no campus.

Quando um pacote chega, o roteador examina o endereço de destino do pacote e verifica a qual sub-rede ele pertence. O roteador pode fazer isso passando o endereço de destino pela operação AND com a máscara para cada sub-rede e verificando se o resultado é o prefixo correspondente. Por exemplo, considere um pacote destinado para o endereço IP 128.208.2.151. Para ver se ele é para o departamento de ciência da computação, realizamos o AND com 255.255.128.0 para apanhar os 17 primeiros bits (que é 128.208.0.0) e vemos se eles correspondem ao endereço do prefixo (que é 128.208.128.0). Eles não correspondem. Verificando os primeiros 18 bits para o departamento de engenharia elétrica, obtemos 128.208.0.0 ao realizar o AND com a máscara de sub-rede. Isso combina com o endereço do prefixo, de modo que o pacote é encaminhado para a interface que leva à rede de engenharia elétrica.

As divisões da sub-rede podem ser mudadas mais tarde, se for preciso, atualizando todas as máscaras de sub-rede nos roteadores dentro da universidade. Como a subdivisão de redes não é visível fora da rede, a alocação de uma nova sub-rede não exige contatar a ICANN nem alterar nenhum banco de dados externo.

CIDR — Classless Inter-Domain Routing

Mesmo que os blocos de endereços IP sejam alocados de modo que os endereços sejam usados de modo eficiente, ainda haverá um problema: a explosão da tabela de roteamento.

Os roteadores nas organizações na borda da rede, como uma universidade, precisam ter uma entrada para cada uma de suas sub-redes, informando ao roteador qual interface usar para chegar a essa rede. Para as rotas até destinos fora da organização, eles podem usar a regra-padrão simples de enviar os pacotes na linha até o ISP que conecta a organização ao restante da Internet. Os outros endereços de destino precisam ser enviados para algum outro lugar.

Os roteadores nos ISPs e backbones no meio da Internet não têm esse luxo. Eles precisam saber que caminho seguir para chegar a cada rede, e nenhum padrão simples funcionará. Esses roteadores de núcleo são considerados como estando na **zona livre padrão** da Internet. Atualmente, ninguém mais sabe realmente quantas redes estão conectadas à Internet, mas é um número grande, provavelmente pelo menos um milhão. Isso pode gerar uma tabela muito grande. Ela pode não parecer grande pelos padrões dos computadores, mas observe que os roteadores precisam realizar uma pesquisa nessa tabela para encaminhar cada pacote, e os roteadores em ISPs grandes podem encaminhar até milhões de pacotes por segundo. Hardware especializado e memória veloz são necessários para processar pacotes nessas velocidades, e não um computador de uso geral.

Além disso, algoritmos de roteamento exigem que cada roteador troque com outros informações sobre os endereços que ele pode alcançar. Quanto maiores as tabelas, mais informações precisam ser comunicadas e processadas. O processamento cresce pelo menos linearmente com o tamanho da tabela. Maior comunicação aumenta a probabilidade de que algumas partes se percam, pelo menos temporariamente, possivelmente levando a instabilidades de roteamento.

O problema da tabela de roteamento poderia ter sido resolvido indo para uma hierarquia mais profunda, como a rede telefônica. Por exemplo, fazer com que cada endereço IP contenha um campo de país, estado/província, cidade, rede e host pode funcionar. Então, cada roteador só precisaria saber como chegar a cada país, estado ou província em seu próprio país, a cidades em seu estado ou província, e a redes em sua cidade. Infelizmente, essa solução exigiria muito mais do que 32 bits para endereços IP e usaria endereços de modo ineficaz (e Liechtenstein teria tantos bits em seus endereços quanto os Estados Unidos).

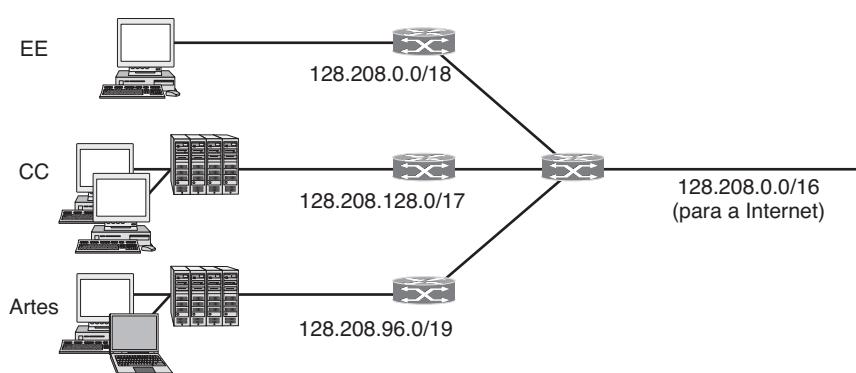



Figura 5.43 | Dividindo um prefixo IP em redes com sub-redes.

Felizmente, há algo que podemos fazer para reduzir os tamanhos da tabela de roteamento. Podemos aplicar a mesma ideia das sub-redes: os roteadores em diferentes locais podem saber a respeito de determinado endereço IP como pertencentes a prefixos de diferentes tamanhos. Porém, em vez de dividir um bloco de endereços em sub-redes, aqui combinamos vários prefixos pequenos em um único prefixo maior. Esse processo é chamado **agregação de rota**. O maior prefixo resultante às vezes é chamado de **super-rede**, ao contrário das sub-redes como a divisão de blocos de endereços.

Com a agregação, os endereços IP estão contidos em prefixos de tamanhos variáveis. O mesmo endereço IP que um roteador trata como parte de um /22 (um bloco contendo 2^{10} endereços) pode ser tratado por outro roteador como parte de um /20 maior (que contém 2^{12} endereços). Fica a cargo de cada roteador ter a informação de prefixo correspondente. Esse projeto funciona com sub-redes e é chamado **CIDR** (**Classless Inter-Domain Routing**). A versão mais recente dele está especificada na RFC 4632 (Fuller e Li, 2006). O nome destaca o contraste com endereços que codificam a hierarquia de classes, que explicaremos em breve.

Para tornar o CIDR mais fácil de entender, vamos considerar um exemplo em que um bloco de 8.192 endereços IP esteja disponível a partir de 194.24.0.0. Suponha que a Universidade de Cambridge precise de 2.048 endereços e tenha recebido os endereços de 194.24.0.0 a 194.24.7.255, junto com a máscara 255.255.248.0. Esse é um prefixo /21. Em seguida, a Universidade de Oxford pede 4.096 endereços. Como um bloco de 4.096 endereços precisa estar em um limite de 4.096 bytes, Oxford não pode receber endereços começando em 194.24.8.0. Em vez disso, ela recebe de 194.24.16.0 a 194.24.31.255, junto com uma máscara de sub-rede 255.255.240.0. Finalmente, a Universidade de Edimburgo pede 1.024 endereços e recebe os endereços de 194.24.8.0 a 194.24.11.255 e a máscara 255.255.252.0. Essas atribuições são resumidas na Tabela 5.6.

Todos os roteadores na zona livre-padrão agora são informados sobre os endereços IP nas três redes. Os roteadores próximos às universidades podem precisar enviar para uma interface de saída diferente para cada um dos prefixos, de modo que eles precisam de uma entrada para cada um dos prefixos em suas tabelas de roteamento. Um exemplo é o roteador em Londres na Figura 5.44.

Agora, vejamos essas três universidades do ponto de vista de um roteador distante em Nova York. Todos os endereços IP nos três prefixos devem ser enviados de Nova York (ou dos Estados Unidos em geral) para Londres. O processo de roteamento em Londres observa isso e combina os três prefixos em uma única entrada agregada para o prefixo 194.24.0.0/19 que ele passa para o roteador em Nova York. Esse prefixo contém 8K endereços e abrange as três universidades e os 1.024 endereços não alocados de outra forma. Usando a agregação, três prefixos foram reduzidos a um, diminuindo os prefixos de que o roteador de Nova York precisa ser informado e as entradas da tabela de roteamento no roteador de Nova York.

Quando a agregação é ativada, esse é um processo automático. Ele depende de quais prefixos estão localizados na Internet, não das ações de um administrador atribuindo endereços às redes. A agregação é bastante usada em toda a Internet e pode reduzir o tamanho das tabelas de roteamento para cerca de 200 mil prefixos.

Outro detalhe é que os prefixos podem se sobrepor. A regra é que os pacotes sejam enviados na direção da rota mais específica, ou do **maior prefixo combinado** que tenha menos endereços IP. O roteamento com o maior prefixo combinado oferece um grau útil de flexibilidade, como pode ser visto no comportamento do roteador em Nova York na Figura 5.45. Esse roteador ainda usa um único prefixo agregado para enviar o tráfego de três universidades para Londres. Porém, o bloco de endereços previamente disponíveis dentro desse prefixo foi agora alocado a uma rede em São Francisco. Uma possibilidade é o roteador em Nova York manter quatro prefixos, enviando pacotes para três deles em Londres e para o quarto em São Francisco. Em vez disso, o roteamento por maior prefixo combinado pode lidar com esse encaminhamento com os dois prefixos que aparecem. Um prefixo geral é usado para o tráfego direto para o bloco inteiro em Londres. Um prefixo mais específico também é usado para direcionar uma parte do prefixo maior para São Francisco. Com a regra do maior prefixo combinado, os endereços IP dentro da rede de São Francisco serão enviados no enlace contínuo para São Francisco, e todos os outros endereços IP no prefixo maior serão enviados para Londres.

Universidade	Primeiro endereço	Último endereço	Quantos	Prefixo
Cambridge	194.24.0.0	194.24.7.255	2.048	194.24.0.0/21
Edimburgo	194.24.8.0	194.24.11.255	1.024	194.24.8.0/22
(Disponível)	194.24.12.0	194.24.15.255	1.024	194.24.12.0/22
Oxford	194.24.16.0	194.24.31.255	4.096	194.24.16.0/20

Tabela 5.6 | Um conjunto de atribuições de endereços IP.

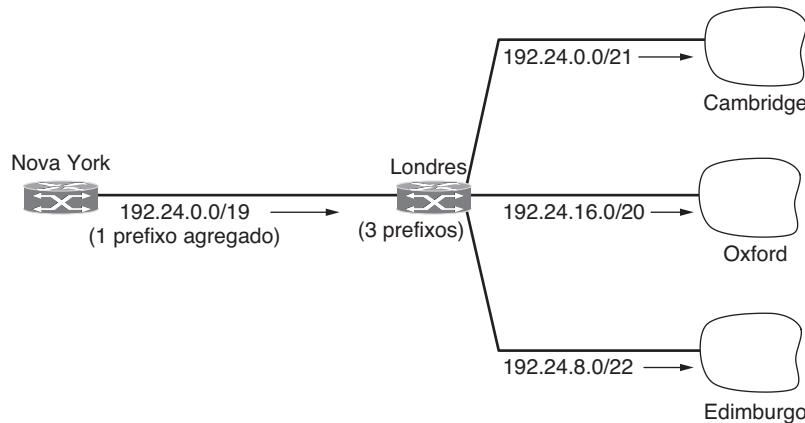



Figura 5.44 | Agregação de prefixos IP.

Por conceito, o CIDR funciona da seguinte forma: quando um pacote chega, a tabela de roteamento é varrida para determinar se o destino se encontra dentro do prefixo. É possível que várias entradas com diferentes tamanhos de prefixo combinem, caso em que é usada a entrada com o maior prefixo. Assim, se houver uma combinação para uma máscara /20 e uma máscara /24, a entrada /24 será usada para pesquisar a interface de saída para o pacote. Porém, esse processo seria tedioso se a tabela fosse realmente varrida entrada por entrada. Em vez disso, foram criados algoritmos complexos para agilizar o processo de combinação de endereço (Ruiz-Sanchez et al., 2001). Os roteadores comerciais utilizam chips VLSI personalizados com esses algoritmos embutidos no hardware.

Endereçamento em classes especiais

Para ajudá-lo a entender melhor por que o CIDR é tão útil, vamos relacionar rapidamente o projeto que o precedeu. Antes de 1993, os endereços IP eram divididos em cinco categorias listadas na Figura 5.46. Essa alocação passou a se chamar **endereçamento em classes**.

Os formatos das classes A, B e C permitem até 128 redes com 16 milhões de hosts cada uma, 16.384 redes com até 65.536 hosts cada uma e 2 milhões de redes (por exemplo, LANs) com até 256 hosts cada uma (embora algumas dessas sejam especiais). Também há suporte para multicast (o formato da classe D), em que um datagrama é direcionado para vários hosts. Os endereços começando com 1111 são reservados para uso futuro. Eles seriam valiosos para usar agora, dada a escassez de espaço de endereços IPv4. Infelizmente, muitos hosts não aceitarão esses endereços como válidos, pois têm ficado fora dos limites por tanto tempo que é difícil ensinar novos truques a hosts antigos.

Esse é um projeto hierárquico, mas, diferentemente do CIDR, os tamanhos dos blocos de endereço são fixos. Existem mais de 2 bilhões de endereços, mas a organização do espaço de endereços por classes desperdiça milhões deles. Em particular, o vilão real é a rede de classe B. Para a maioria das organizações, uma rede de classe A, com 16 milhões de endereços, é muito grande, e uma rede de classe C, com 256 endereços, é muito pequena. Uma rede de classe B, com 65.536, é ideal. No folclore da Internet, essa situação é conhecida como o **problema dos três ursos** [como em *Goldilocks and the Three Bears* (Southey, 1848)].

Na realidade, porém, um endereço de classe B é muito grande para a maioria das organizações. Os estudos têm mostrado que mais da metade de todas as redes de classe B tem menos de 50 hosts. Uma rede de classe C teria sido suficiente, mas, sem dúvida, cada organização que pediu um endereço de classe B pensou que um dia ultrapassaria o campo de host de 8 bits. Fazendo um retrospecto, poderia ter sido melhor ter tido redes de classe C usando 10 bits em vez de 8 para o campo de host, permitindo 1.022 hosts por rede. Se isso acontecesse, a maioria das organizações provavelmente teria ficado com uma rede de classe C e haveria meio milhão delas (contra apenas 16.384 redes de classe B).

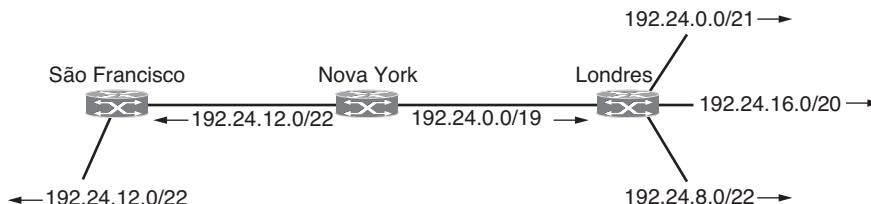



Figura 5.45 | Roteamento pelo maior prefixo combinado no roteador em Nova York.

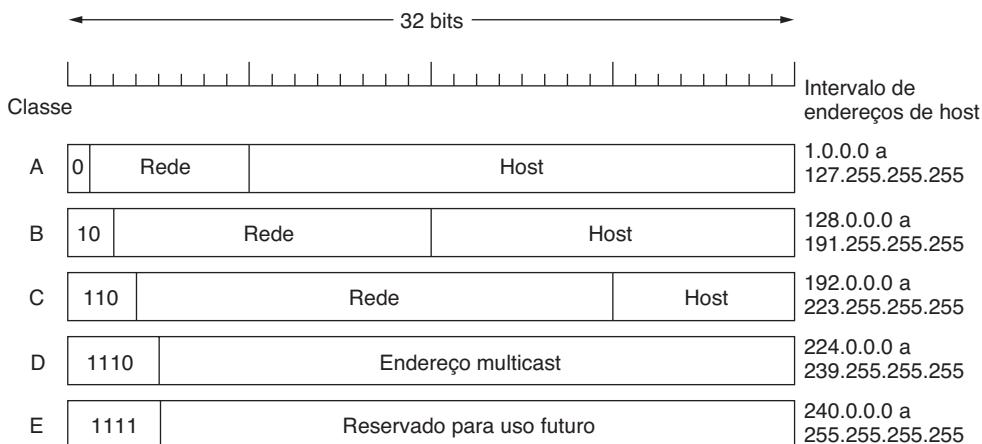



Figura 5.46 | Formatos de endereços IP.

É difícil culpar os projetistas da Internet por não ter oferecido mais (e menores) endereços de classe B. Na época em que foi feita a decisão de criar as três classes, a Internet era uma rede de pesquisa conectando as principais universidades de pesquisa dos Estados Unidos (mais algumas poucas empresas e militares realizando pesquisas com redes). Ninguém percebeu que a Internet se tornava um sistema de comunicação de mercado em massa, competindo com a rede telefônica. Na época, alguém sem dúvida disse: "Os Estados Unidos têm cerca de 2 mil faculdades e universidades. Mesmo que todas elas se conectem à Internet e muitas outras universidades em outros países também se juntem, nunca chegaremos aos 16 mil, pois não existem tantas universidades no mundo inteiro. Além do mais, ter como número de host um número inteiro de bytes agiliza o processamento do pacote" (o que, na época, era feito inteiramente no software). Talvez algum dia as pessoas voltem e culpem aqueles que projetaram o esquema do número de telefone e digam: "Que idiotas. Por que eles não incluíram o número do planeta no número do telefone?". Porém, no momento, isso parece não ser necessário.

Para enfrentar esses problemas, as sub-redes foram introduzidas para atribuir blocos de endereços com flexibilidade dentro de uma organização. Mais tarde, o CIDR foi acrescentado para reduzir o tamanho da tabela de roteamento global. Hoje, os bits que indicam se um endereço IP pertence à rede de classe A, B ou C não são mais usados, embora as referências a essas classes na literatura ainda sejam comuns.

Para ver como o descarte das classes tornou o encaminhamento mais complicado, considere como isso era simples no sistema antigo com classes. Quando um pacote chegava a um roteador, uma cópia do endereço IP era deslocada 28 bits para a direita, para gerar um número de classe de 4 bits. Um desvio de 16 bits desviava então os pacotes para as classes A, B, C (e D, E), com oito dos casos para a classe A, quatro para a classe B e dois para a classe C. Então, o código para cada classe mascarava o número de rede de 8, 16 ou 24 bits e o alinhava à direita em uma palavra de 32 bits. O número de rede era então pesquisado na tabela A, B ou C, normalmente indexando para redes A e B e realizando o hash para redes C. Quando a entrada era encontrada, a interface de saída poderia ser pesquisada e o pacote, encaminhado. Isso é muito mais simples do que a operação de combinação de maior prefixo combinado, que não pode mais usar uma simples pesquisa de tabela, pois um endereço IP pode ter um prefixo de qualquer tamanho.

Os endereços de classe D continuam a ser usados na Internet para multicast. Na realidade, pode ser mais preciso dizer que eles estão começando a ser usados para multicast, pois o multicast na Internet não foi muito implementado no passado.

Também existem vários outros endereços que possuem significados especiais, como mostra a Figura 5.47. O endereço IP 0.0.0.0, o menor deles, é usado pelos hosts quando estão sendo inicializados. Isso significa "essa rede" ou "esse host". Os endereços IP com 0 como número de rede referem-se à rede atual. Esses endereços permitem que as máquinas se refiram à sua própria rede sem conhecer seu número (mas elas precisam conhecer a máscara de rede para saber quantos 0s incluir). O endereço que consiste apenas em 1s, ou 255.255.255.255 — o mais alto —, é usado para apontar todos os hosts na rede indicada. Ele permite o broadcasting na rede local, normalmente uma LAN. Os endereços com um número de rede apropriado e apenas 1s no campo de host permitem que as máquinas enviem pacotes de broadcast para LANs distantes em qualquer parte da Internet. Porém, muitos administradores de rede desativam esse recurso, pois ele é um sério risco à segurança. Finalmente, todos os endereços na forma 127.*xx.yy.zz* são reservados para o teste de loopback. Os pacotes enviados a esse endereço não são enviados para os fios; eles são processados localmente e tratados como pacotes de chegada. Isso permite que os pacotes sejam enviados para o host sem que um transmissor conheça seu número, o que é útil para fins de teste.

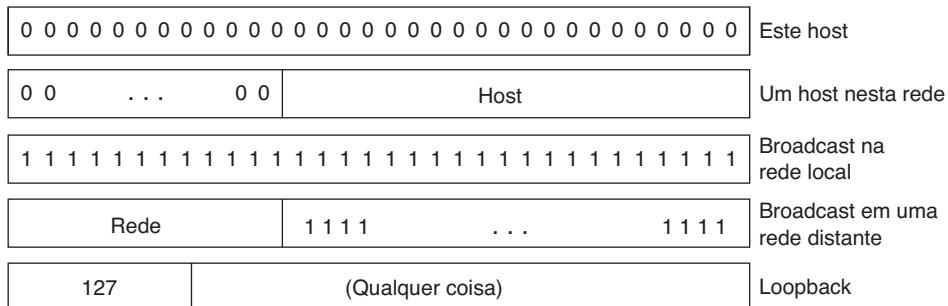



Figura 5.47 | Endereços IP especiais.

NAT – Network Address Translation

Os endereços IP são escassos. Um ISP poderia ter um endereço /16, fornecendo 65.534 números de hosts. Se ele tiver um número maior do que esse de clientes, haverá um problema.

Essa escassez levou à criação de técnicas para usar o endereço IP com cautela. Uma técnica é atribuir dinamicamente um endereço IP ao computador quando ele se conectar e usar a rede, tomando-o de volta quando o host se tornar inativo. Desse modo, um único endereço /16 poderá manipular até 65.534 usuários ativos.

Essa estratégia funciona bem em alguns casos, por exemplo, para redes discadas e computadores móveis e outros que podem estar temporariamente ausentes ou desligados. Porém, ela não funciona muito bem para clientes empresariais. Muitos PCs nas empresas devem estar ligados continuamente. Alguns são máquinas de funcionários que recebem backup à noite, e algumas são servidores que podem ter de atender a uma solicitação remota a qualquer momento, sem aviso. Essas empresas têm um enlace de acesso que sempre oferece conectividade ao restante da Internet.

Cada vez mais, essa situação também se aplica a usuários domésticos que assinam serviços de ADSL ou Internet por cabo, pois não existe cobrança por conexão (apenas uma taxa fixa mensal). Muitos desses usuários têm dois ou mais computadores em casa, muitas vezes um para cada membro da família, e todos eles querem estar on-line o tempo todo. A solução aqui é conectar todos os computadores a uma rede doméstica por meio de uma LAN e inserir um roteador (*wireless*) nela. O roteador, então, se conecta ao ISP. Do ponto de vista do ISP, agora a família equivale a uma pequena empresa com alguns computadores. Bem-vindos à Silva & Silva Ltda. Com as técnicas que vimos até aqui, cada computador precisa ter seu próprio endereço IP o dia inteiro. Para um ISP com muitos milhares de clientes, principalmente clientes empresariais e famílias que são como pequenas empresas, a demanda por endereços IP pode ultrapassar rapidamente o bloco disponível.

O problema de esgotar os endereços IP não é um problema teórico que poderia ocorrer em algum momento no futuro distante. Ele está acontecendo aqui e agora. A solução a longo prazo é a Internet inteira migrar para o IPv6, que tem endereços de 128 bits. Essa transição está ocorrendo com lentidão e a conclusão do processo demorará muitos anos. Para contornar a situação nesse meio-tempo, foi necessário fazer uma rápida correção. Essa correção veio sob a forma da **NAT** (**network address translation**), descrita na RFC 3022 e que resumiremos a seguir. Para obter informações adicionais, consulte Dutcher (2001).

A ideia básica por trás do NAT é atribuir a cada empresa um único endereço IP (ou, no máximo, um número pequeno deles) para tráfego na Internet. *Dentro* da empresa, todo computador obtém um endereço IP exclusivo, usado para roteamento do tráfego interno. Porém, quando um pacote sai da empresa e vai para o ISP, ocorre uma conversão do endereço IP interno para o endereço IP público. Essa tradução utiliza três intervalos de endereços IP que foram declarados como privativos. As redes podem utilizá-los internamente como desejarem. A única regra é que nenhum pacote contendo esses endereços pode aparecer na própria Internet. Os três intervalos reservados são:

10.0.0.0 – 10.255.255.255/8 (16.777.216 hosts)
172.16.0.0 – 172.31.255.255/12 (1.048.576 hosts)
192.168.0.0 – 192.168.255.255/16 (65.536 hosts)

O primeiro intervalo permite a utilização de 16.777.216 endereços (com exceção dos endereços contendo apenas 0 ou apenas 1, como sempre) e é a escolha habitual, mesmo que a rede não necessite de tantos endereços.

A operação do NAT é mostrada na Figura 5.48. Dentro das instalações do cliente, toda máquina tem um endereço exclusivo da forma 10.*x*.*y*.*z*. Porém, quando um pacote deixa as instalações da empresa, ele passa por um **NAT** que converte o endereço de origem IP interno, 10.0.0.1 na figura, no endereço IP verdadeiro da empresa, 198.60.42.12 nesse exemplo. Com frequência, o NAT é combinado em um único dispositivo com um firewall, que oferece segurança por meio do controle cuidadoso do que entra na empresa e do que sai dela. Estudaremos os firewalls no Capítulo 8. Também é possível integrar a NAT a um roteador ou modem ADSL.

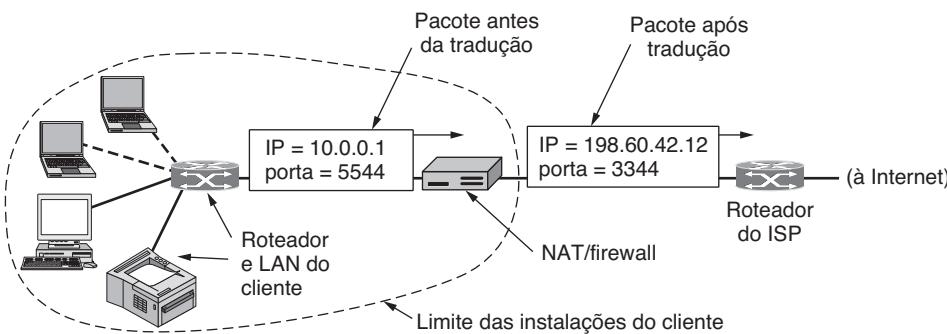



Figura 5.48 | Posicionamento e operação de um NAT.

Até agora, deixamos de lado um pequeno detalhe: quando a resposta volta (por exemplo, de um servidor da Web), ela é naturalmente endereçada para 198.60.42.12; então, como o NAT sabe por qual endereço deve substituir o endereço da resposta? Aqui reside o problema com a NAT: se houvesse um campo sobressalente no cabeçalho IP, ele poderia ser usado para controlar qual foi o transmissor real, mas só resta um bit ainda não utilizado. Em princípio, uma nova opção poderia ser criada para conter o endereço de origem verdadeiro, mas isso exigiria mudar o código do IP em todas as máquinas na Internet inteira para manipular a nova opção. Essa não é uma alternativa promissora para uma correção rápida.

O que realmente aconteceu é descrito a seguir. Os projetistas do NAT observaram que a maioria dos pacotes IP transporta uma carga útil TCP ou UDP. Quando estudarmos o TCP e o UDP no Capítulo 6, veremos que ambos têm cabeçalhos contendo uma porta de origem e uma de destino. Descreveremos a seguir apenas as portas TCP, mas o mesmo princípio é válido para as portas UDP. As portas são inteiros de 16 bits que indicam onde a conexão TCP começa e termina. Essas portas fornecem o campo necessário para fazer o NAT funcionar.

Quando um processo deseja estabelecer uma conexão TCP com um processo remoto, ele se associa a uma porta TCP não utilizada em sua própria máquina, a qual é chamada **porta de origem** e informa ao código do TCP para onde devem ser enviados os pacotes que chegarem pertencentes a essa conexão. O processo também fornece uma **porta de destino** para informar a quem devem ser entregues os pacotes no lado remoto. As portas de 0 a 1023 são reservadas para serviços conhecidos. Por exemplo, a porta 80 é usada por servidores da Web, de forma que clientes remotos possam localizá-los. Cada mensagem TCP enviada contém uma porta de origem e uma de destino. Juntas, elas servem para identificar os processos que utilizam a conexão nas duas extremidades.

Uma analogia deve tornar mais claro o uso das portas. Imagine uma empresa com um único número de telefone principal. Quando as pessoas ligam para o número principal, acessam um telefonista, que pergunta qual ramal elas desejam e em seguida as conecta a esse ramal. O número principal é equivalente ao endereço IP do cliente, e os ramais em ambas as extremidades são equivalentes às portas. As portas constituem um grupo extra de 16 bits de endereçamento que identifica o processo que receberá cada pacote de entrada.

Usando o campo *Porta de origem*, podemos resolver nosso problema de mapeamento. Sempre que um pacote de saída entra no NAT, o endereço de origem 10.*x.y.z* é substituído pelo endereço IP verdadeiro do cliente. Além disso, o campo *Porta de origem* do TCP é substituído por um índice na tabela de tradução de 65.536 entradas do NAT. Essa entrada da tabela contém a porta de origem e o endereço IP original. Por fim, tanto o checksum do cabeçalho IP quanto o do cabeçalho TCP são recalculados e inseridos no pacote. É necessário substituir o campo *Porta de origem*, pois as conexões das máquinas 10.0.0.1 e 10.0.0.2 podem, por exemplo, usar a mesma porta 5000, e, assim, o campo *Porta de origem* sozinho não é suficiente para identificar o processo transmissor.

Quando um pacote chega ao NAT vindo do ISP, o campo *Porta de origem* no cabeçalho de TCP é extraído e usado como índice para a tabela de mapeamento do NAT. A partir da entrada localizada, o endereço IP interno e o campo *Porta de origem* do TCP original são extraídos e inseridos no pacote. Em seguida, os checksums do IP e do TCP são recalculados e inseridos no pacote. O pacote é, então, repassado ao roteador do cliente para entrega normal, utilizando o endereço 10.*x.y.z*.

Embora esse tipo de esquema resolva o problema, muitas pessoas na comunidade IP o consideram uma abominação. Em resumo, a seguir estão algumas objeções. Primeiro, o NAT viola o modelo arquitetônico do IP, que estabelece que todo endereço IP identifica de forma exclusiva uma única máquina em todo o mundo. Toda a estrutura de software da Internet se baseia nesse fato. Com o NAT, milhares de máquinas podem usar (e usam) o endereço 10.0.0.1.

Em segundo lugar, o NAT fere o modelo de conectividade de ponto a ponto da Internet, que diz que qualquer host pode enviar um pacote para outro a qualquer momento. Como o mapeamento no NAT é configurado por pacotes

de saída, os pacotes de entrada não podem ser aceitos antes dos de saída. Na prática, isso significa que um usuário doméstico com um NAT pode fazer conexões TCP/IP com um servidor Web remoto, mas um usuário remoto não pode fazer conexões com um servidor de jogos na rede doméstica com um NAT. É necessário usar técnicas de configuração especiais ou de **travessia do NAT** para dar suporte a esse tipo de situação.

Terceiro, o NAT faz a Internet mudar suas características de rede não orientada a conexões para uma espécie de rede orientada a conexões. O problema é que o NAT deve manter informações (o mapeamento) para cada conexão que passa por ela. Manter o estado da conexão é uma propriedade das redes orientadas a conexões, e não das redes não orientadas a conexões. Se o NAT sofrer uma pane e sua tabela de mapeamento se perder, todas as conexões TCP serão destruídas. Na ausência do NAT, panes em roteadores não terão nenhum efeito sobre o TCP. O processo transmissor simplesmente entrará em timeout dentro de alguns segundos e retransmitirá todos os pacotes não confirmados. Com o NAT, a Internet se torna tão vulnerável quanto uma rede comutada por circuitos.

Em quarto lugar, o NAT viola a regra mais fundamental da distribuição de protocolos em camadas: a camada k não pode fazer nenhuma suposição sobre o que a camada $k + 1$ inseriu no campo de carga útil. Esse princípio básico existe para manter as camadas independentes. Se o TCP for atualizado mais tarde para TCP-2, com um layout de cabeçalho diferente (por exemplo, portas de 32 bits), o NAT falhará. Toda a ideia de protocolos em camadas tem o objetivo de assegurar que as mudanças em uma camada não exigirão mudanças em outras. O NAT destrói essa independência.

Quinto, os processos na Internet não são obrigados a usar o TCP ou o UDP. Se um usuário na máquina *A* decidir empregar algum novo protocolo de transporte para se comunicar com um usuário na máquina *B* (por exemplo, no caso de uma aplicação de multimídia), a introdução de um NAT fará a aplicação falhar, porque o NAT não será capaz de localizar corretamente o campo *Porta de origem* do TCP.

Um sexto problema relacionado é que algumas aplicações usam várias conexões TCP/IP ou portas UDP de maneiras predefinidas. Por exemplo, o **protocolo de transferência de arquivos** padrão, o **FTP** (**File Transfer Protocol**), insere endereços IP no corpo do pacote para o receptor extrair e usar. Por não saber nada sobre esses arranjos, o NAT não pode reescrever os endereços IP ou considerá-los de alguma outra forma. Essa falta de conhecimento significa que o FTP e outras aplicações, como o protocolo de telefonia da Internet H.323 (que estudaremos no Capítulo 7), têm essa propriedade e podem falhar na presença da NAT, a menos que sejam tomadas precauções especiais. Talvez seja possível corrigir o NAT para esses casos, mas ter de corrigir o código no NAT toda vez que surge uma nova aplicação não é uma boa ideia.

Por fim, como o campo *Porta de origem* do TCP é de 16 bits, no máximo 65.536 máquinas podem ser mapeadas em um endereço IP. Na realidade, o número é um pouco menor, porque as primeiras 4.096 portas estão reservadas para usos especiais. Porém, se vários endereços IP estiverem disponíveis, cada um deles poderá manipular até 61.440 máquinas.

Esses e outros problemas com o NAT são discutidos na RFC 2993. Apesar dos problemas, o NAT é muito usado na prática, especialmente em redes domésticas e de pequenas empresas, como a última técnica ágil para lidar com a falta de endereços IP. Ele tem sido usado com firewalls para privacidade, pois impede os pacotes de chegada não solicitados, por padrão. Por esse motivo, é provável que ela não desapareça mesmo quando o IPv6 tiver sido implantado de modo geral.

5.6.3 IP Versão 6

O IP tem sido muito usado há décadas. Ele tem funcionado extremamente bem, conforme demonstrado pelo crescimento exponencial da Internet. Infelizmente, ele tornou-se vítima de sua própria popularidade: está próximo de esgotar os endereços disponíveis. Até mesmo com CIDR e NAT usando endereços com mais cautela, os últimos endereços IPv4 deverão ser atribuídos pela ICANN antes do final de 2012. Esse desastre iminente foi reconhecido há quase duas décadas e gerou muita discussão e controvérsia dentro da comunidade da Internet sobre o que fazer a respeito.

Nesta seção, descreveremos o problema e várias soluções propostas. A única solução a longo prazo é passar para endereços maiores. O **IPv6** (**IP versão 6**) é um projeto substituto que faz exatamente isso. Ele usa endereços de 128 bits; uma escassez desses endereços provavelmente não ocorrerá no futuro previsível. Porém, o IPv6 provou ser muito difícil de implementar. Ele é um protocolo diferente da camada de rede e não se interliga realmente com o IPv4, apesar de muitas semelhanças. Além disso, as empresas e usuários não sabem ao certo por que eles deveriam querer o IPv6 de qualquer forma. O resultado é que o IPv6 está implementado e é usado em apenas uma pequena fração da Internet (estima-se que seja 1 por cento), apesar de ser um Internet Standard desde 1998. Os próximos anos serão interessantes, pois os poucos endereços IPv4 restantes serão alocados. Será que as pessoas começarão a leiloar seus endereços IPv4 no eBay? Será que surgirá um mercado negro para isso? Quem sabe?

Além desses problemas técnicos, há uma outra questão em paralelo. No início, a Internet era amplamente usada por universidades, indústrias de alta tecnologia e órgãos governamentais dos Estados Unidos (especialmente pelo Departamento de Defesa). Com a explosão da Internet a partir de meados da década de 1990, ela começou a ser usada por um grupo diferente de pessoas, em especial as com

necessidades específicas. Por um lado, milhares de pessoas com smartphones a utilizam para manter contato com suas bases. Por outro, com a inevitável convergência das indústrias de informática, comunicação e entretenimento, talvez não demore para que cada telefone e cada televisor do mundo seja um nó da Internet, resultando no uso de áudio e vídeo por demanda em um bilhão de máquinas. Sob essas circunstâncias, ficou claro que o IP precisava evoluir para se tornar mais flexível.

Vendo esses problemas no horizonte, em 1990 a IETF começou a trabalhar em uma nova versão do IP, que nunca ficaria sem endereços, resolveria uma série de outros problemas e também seria mais flexível e eficiente. Seus principais objetivos eram:

1. Aceitar bilhões de hosts, mesmo com alocação ineficiente de espaço de endereços.
2. Reduzir o tamanho das tabelas de roteamento.
3. Simplificar o protocolo, de modo a permitir que os roteadores processem os pacotes com mais rapidez.
4. Oferecer mais segurança (autenticação e privacidade).
5. Dar mais importância ao tipo de serviço, particularmente no caso de dados em tempo real.
6. Auxiliar o multicasting, possibilitando a especificação de objetivos.
7. Permitir que um host mude de lugar sem precisar mudar de endereço.
8. Permitir que o protocolo evoluísse no futuro.
9. Permitir a coexistência entre protocolos novos e antigos durante anos.

O projeto do IPv6 apresentou uma oportunidade importante para melhorar todos os recursos no IPv4 que ficaram aquém do que se deseja agora. Para chegar a um protocolo que atendesse a todos esses requisitos, a IETF convocou os interessados em apresentar suas propostas na RFC 1550. Inicialmente, foram recebidas 21 respostas. Em dezembro de 1992, havia sete propostas muito interessantes em estudo. As propostas variavam desde pequenos ajustes no IP até sua eliminação pura e simples, com a criação de um protocolo totalmente diferente.

Uma proposta era executar o TCP sobre o CLNP, o protocolo de camada de rede desenvolvido para OSI. Com seus endereços de 160 bits, o CLNP seria capaz de oferecer um espaço de endereços infinito, pois poderia dar a cada molécula de água nos oceanos endereços suficientes (cerca de 2^5) para montar uma pequena rede. Essa escolha também unificaria os dois principais protocolos da camada de rede. No entanto, para muita gente, isso seria o mesmo que admitir que o mundo OSI ainda tinha suas vantagens, uma afirmação politicamente incorreta nos círculos da Internet. A padronização do protocolo CLNP tinha características muito parecidas com a do IP; portanto, não podemos afirmar que os dois protocolos sejam, de fato, muito diferentes. Na verdade, o protocolo que acabou sendo escolhido apresenta muito mais diferenças em relação ao IP do que o CLNP. Um dos fatores que pesaram contra o CLNP foi a baixa qualidade em relação aos tipos de serviços oferecidos, algo de fundamental importância para uma transmissão multimídia eficiente.

As três melhores propostas foram publicadas na *IEEE Network* (Deering, 1993; Francis, 1993; e Katz e Ford, 1993). Depois de muita discussão, revisão e disputa, foi selecionada uma versão combinada modificada das propostas de Deering e Francis, agora chamada **SIPP** (**Simple Internet Protocol Plus**), à qual foi atribuída a designação **IPv6**.

O IPv6 atende a todos os objetivos da IETF, preservando os bons recursos do IP, descartando ou reduzindo a importância das características ruins e criando outras quando necessário. Genericamente, o IPv6 não é compatível com o IPv4, mas o é com todos os outros protocolos auxiliares da Internet, incluindo TCP, UDP, ICMP, IGMP, OSPF, BGP e DNS, apesar de serem necessárias pequenas modificações para lidar com endereços mais longos. Os principais recursos do IPv6 serão discutidos a seguir. Para obter mais informações sobre ele, consulte as RFCs de 2460 a 2466.

Em primeiro lugar, o IPv6 tem endereços mais longos que o IPv4. Eles têm 128 bits, o que resolve o problema que o IPv6 se propõe a resolver: oferecer um número praticamente ilimitado de endereços na Internet. Voltaremos a descrever os endereços mais adiante.

O segundo aperfeiçoamento importante no IPv6 é a simplificação do cabeçalho. Ele contém apenas sete campos (contra os 13 do IPv4). Essa mudança permite aos roteadores processar os pacotes com mais rapidez e, dessa forma, melhorar o throughput e o atraso. Também voltaremos a descrever o cabeçalho em breve.

A terceira mudança importante foi o melhor suporte para as opções oferecidas. Essa mudança era essencial para o novo cabeçalho, pois os campos então obrigatórios agora são opcionais. Além disso, é diferente a forma como as opções são representadas, o que torna mais simples para os roteadores ignorar as opções a que eles não se propõem. Esse recurso diminui o tempo de processamento de pacotes.

Uma quarta área em que o IPv6 representa um grande avanço é a segurança. A IETF já estava farta de ver reportagens nos jornais com meninos precoces de 12 anos que, utilizando seus computadores pessoais, conseguiam devassar segredos de grandes instituições financeiras e militares pela Internet. Havia uma forte sensação de que era preciso fazer algo para melhorar a segurança. A autenticação e a privacidade são recursos importantes do novo IP. Porém, mais tarde essas características foram integradas ao IPv4; assim, na área de segurança não há mais diferenças tão grandes.

Por fim, foi dada maior atenção à qualidade de serviço. Diversos esforços corajosos foram feitos no passado para melhorar a QoS; porém, com o crescimento atual da multimídia na Internet, a sensação de urgência é maior.

O cabeçalho principal do IPv6

O cabeçalho do IPv6 é mostrado na Figura 5.49. O campo *Versão* é sempre 6 para o IPv6 (e 4 para o IPv4). Durante o período de transição do IPv4, que já passou de uma década, os roteadores serão capazes de examinar esse campo para identificar o tipo de pacote que eles têm. A propósito, a realização desse teste desperdiça algumas instruções no caminho crítico, visto que o cabeçalho do enlace de dados normalmente indica o protocolo de rede para demultiplexação e, portanto, alguns roteadores podem pular a verificação. Por exemplo, o campo *Tipo* da Ethernet tem diferentes valores para indicar uma carga útil IPv4 ou IPv6. As discussões entre os que desejam "fazer o que é certo" e os que querem "tornar o processo mais rápido" ainda deverão se estender por um longo tempo e serão motivo de muita polêmica.

O campo *Serviços diferenciados* (originalmente chamado *Classe de tráfego*) é usado para distinguir a classe de serviço para pacotes com diferentes requisitos de entrega em tempo real. Ele é usado com a arquitetura de serviço diferenciado para qualidade de serviço da mesma maneira que o campo de mesmo nome no pacote IPv4. Além disso, os 2 bits de baixa ordem são usados para sinalizar indicações explícitas de congestionamento, novamente da mesma maneira que no IPv4.

O campo *Rótulo de fluxo* permite que uma origem e um destino marquem grupos de pacotes que têm os mesmos requisitos e devem ser tratados da mesma maneira pela rede, formando uma pseudoconexão. Por exemplo, um fluxo de pacotes entre um processo de determinado host de origem e certo processo de um host de destino específico pode ter severas restrições em termos de atraso e, por essa razão, necessitar de uma largura de banda reservada. O fluxo pode ser configurado com antecedência e ter um identificador atribuído a ele. Quando aparece um pacote com o campo *Rótulo de fluxo* com valor diferente de zero, todos os roteadores podem verificar nas tabelas internas que tipo de tratamento especial ele exige. Na prática, os fluxos são uma tentativa de ter a flexibilidade de uma rede de datagramas com as garantias de uma rede de circuitos virtuais.

Para fins de qualidade de serviço, cada fluxo é designado por endereço de origem, endereço de destino e número de fluxo. Esse projeto significa que até 2^{20} fluxos podem estar ativos ao mesmo tempo entre determinado par de endereços IP. Por essa razão, quando dois fluxos enviados por diferentes hosts e com o mesmo número de fluxo passarem pelo mesmo roteador, este será capaz de distingui-los usando os endereços de origem e de destino. Para que os roteadores possam analisar os números de fluxo com mais facilidade, eles serão escolhidos ao acaso, em vez de ser atribuídos sequencialmente a partir de 1.

O campo *Tamanho de carga útil* determina o número de bytes que seguem o cabeçalho de 40 bytes da Figura 5.49. O nome desse campo, que no IPv4 era *Tamanho total*, foi alterado em virtude de uma pequena mudança de significado a que foi submetido: os 40 bytes do cabeçalho deixaram de ser contados como parte do tamanho, como acontecia até então. Essa mudança significa que a carga útil agora pode ser de 65.535 bytes em vez de meros 65.515 bytes.

O campo *Próximo cabeçalho* revela um segredo. O cabeçalho pode ser simplificado, pois existe a possibilidade de haver outros cabeçalhos de extensão (opcionais). Esse campo informa quais dos seis cabeçalhos de extensão (atuais) seguem esse cabeçalho, se houver algum. Se esse cabeçalho for o último do IP, o campo *Próximo cabeçalho* revelará para qual tratador de protocolo de transporte (por exemplo, TCP, UDP) o pacote deverá ser enviado.

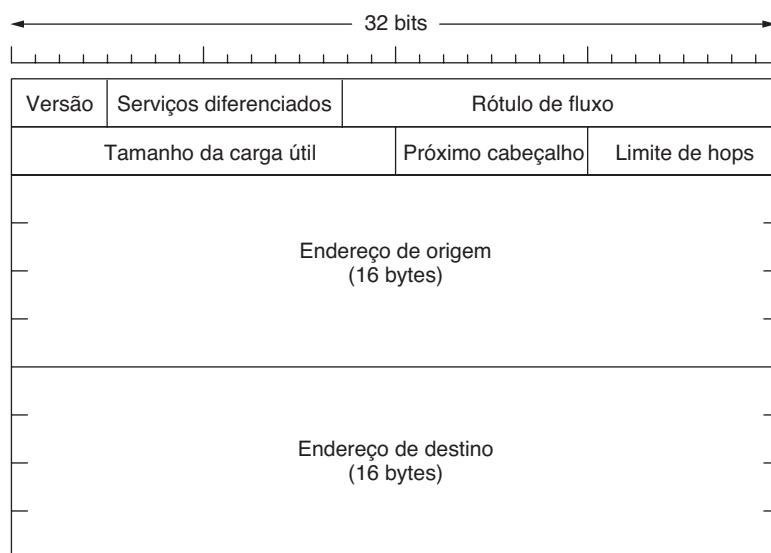



Figura 5.49 | O cabeçalho IPv6 fixo (obrigatório).

O campo *Limite de hops* é usado para impedir que os pacotes tenham duração eterna. Na prática, ele é igual ao campo *TTL* do IPv4, ou seja, um campo que é decrementado a cada hop. Teoricamente, no IPv4 ele denotava um tempo em segundos, mas nenhum roteador o utilizava dessa maneira. Por esse motivo, seu nome foi alterado para refletir o modo como ele é usado de fato.

Em seguida, vêm os campos *Endereço de origem* e *Endereço de destino*. A proposta original de Deering, o SIP, utilizava endereços de 8 bytes; porém, durante o processo de revisão, muitas pessoas perceberam que, com endereços de 8 bytes, o IPv6 esgotaria os endereços disponíveis em apenas algumas décadas, enquanto os endereços de 16 bytes nunca se esgotariam. Outras pessoas afirmavam que 16 bytes seriam um exagero, e outras alegavam que endereços de 20 bytes seriam compatíveis com o protocolo de datagramas do OSI. Ainda outra facção queria endereços de tamanho variável. Depois de muita discussão, chegou-se à conclusão de que a melhor opção era utilizar endereços de 16 bytes.

Foi criada uma nova notação para representar endereços de 16 bytes. Eles são escritos sob a forma de oito grupos de quatro dígitos hexadecimais, separados por sinais de dois-pontos entre os grupos, como no exemplo a seguir:

8000:0000:0000:0000:0123:4567:89AB:CDEF

Tendo em vista que vários endereços conterão muitos zeros, foram autorizadas três otimizações. Em primeiro lugar, os zeros à esquerda dentro de um grupo podem ser omitidos, de modo que 0123 possa ser escrito como 123. Em segundo lugar, um ou mais grupos de 16 bits zero podem ser substituídos por um par de sinais de dois-pontos. Consequentemente, o endereço anterior pode ser escrito da seguinte maneira:

8000::123:4567:89AB:CDEF

Por fim, os endereços IPv4 podem ser escritos empregando-se um par de sinais de dois-pontos e um número decimal tradicional, como neste exemplo:

::192.31.20.46

Talvez não seja necessário ser tão explícito em relação a isso, mas existem muitos endereços de 16 bytes. Especificamente, existem 2^{128} deles, o que significa cerca de 3×10^{38}. Se o planeta inteiro, terra e água, fosse coberto de computadores, o IPv6 permitiria 7×10^{23} endereços IP por metro quadrado. Os estudantes de química perceberão que esse número é maior que o número de Avogadro. Embora não exista a intenção de dar a cada molécula na superfície da Terra seu próprio endereço IP, não estamos longe de chegar a essa marca.

Na prática, o espaço de endereços não será usado com eficiência, exatamente como acontece com o espaço de endereços dos números telefônicos (o código de área de Manhattan, 212, está próximo da saturação, mas o de Wyoming, 307, está quase vazio). Na RFC 3194, Durand e Huitema calcularam que, usando a alocação dos números de telefones como um guia, mesmo considerando-se a hipótese mais pessimista, ainda assim haverá mais de mil endereços IP por metro quadrado de toda a superfície da Terra (incluindo rios e mares). Em qualquer situação provável, haverá trilhões deles por metro quadrado. Em resumo, parece improvável que eles venham a se esgotar em qualquer ponto no futuro previsível.

É interessante comparar o cabeçalho do IPv4 (Figura 5.41) com o cabeçalho do IPv6 (Figura 5.49) e ver o que foi mantido e o que foi descartado no IPv6. O campo *IHL* foi eliminado, pois o cabeçalho do IPv6 tem um tamanho fixo. O campo *Protocolo* foi retirado porque o campo *Próximo cabeçalho* identifica o que vem depois do último cabeçalho IP (por exemplo, um segmento UDP ou TCP).

Todos os campos relacionados à fragmentação foram removidos, pois o IPv6 dá um tratamento diferente à fragmentação. Para começar, todos os hosts e roteadores compatíveis com o IPv6 devem determinar dinamicamente o tamanho de pacote que será usado. Eles fazem isso usando o procedimento de descoberta da MTU do caminho, descrito na Seção 5.5.5. Resumindo, quando um host enviar um pacote IPv6 muito grande, em vez de fragmentá-lo, o roteador incapaz de encaminhá-lo descartará o pacote e enviará uma mensagem de erro de volta ao host transmissor. Essa mensagem instrui o host a dividir todos os novos pacotes enviados a esse destino. Fazer o host enviar pacotes que têm o tamanho correto em primeiro lugar é muito mais eficiente do que fazer com que os roteadores os fragmentem no ato. Além disso, o pacote de tamanho mínimo que os roteadores precisam saber encaminhar aumentou de 576 para 1.280 bytes, permitindo 1.024 bytes de dados e muitos cabeçalhos.

Por fim, o campo *Checksum* foi eliminado, porque esse cálculo reduz o desempenho de forma significativa. Com as redes confiáveis usadas atualmente, além do fato de a camada de enlace de dados e as camadas de transporte terem seus próprios checksums, a importância de um novo checksum é insignificante, se comparada com a queda de desempenho que ele implica. Com a remoção de todos esses recursos, o protocolo da camada de rede ficou muito mais enxuto e prático. Assim, o objetivo do IPv6 — um protocolo a um só tempo rápido e flexível, capaz de oferecer um grande espaço de endereços — foi atendido por esse projeto.

Cabeçalhos de extensão

Ocasionalmente, alguns dos campos ausentes do IPv4 ainda serão necessários; assim, o IPv6 introduziu o conceito de **cabeçalho de extensão** (opcional). Esses cabeçalhos podem ser criados com a finalidade de oferecer informações extras, desde que elas sejam codificadas de maneira eficiente. Atualmente, há seis tipos de cabeçalhos de extensão definidos, mostrados na Tabela 5.7. Todos eles são opcionais, mas, se houver mais de um, eles terão de aparecer logo depois do cabeçalho fixo e, de preferência, na ordem listada.

Cabeçalho de extensão	Descrição
Hop-by-hop options	Informações diversas para os roteadores
Destination options	Informações adicionais para o destino
Routing	Lista parcial de roteadores a visitar
Fragmentation	Gerenciamento de fragmentos de datagramas
Authentication	Verificação da identidade do transmissor
Encrypted security payload	Informações sobre o conteúdo criptografado

Tabela 5.7 | Cabeçalhos de extensão do IPv6.

Alguns desses cabeçalhos têm um formato fixo; outros contêm um número variável de campos de comprimento variável. Nesses casos, cada item é codificado como uma tupla (*Tipo*, *Tamanho*, *Valor*). *Tipo* é um campo de 1 byte que identifica a opção. Os valores de *Tipo* foram escolhidos de tal forma que os dois primeiros bits informem o que os roteadores que não sabem como processar a opção devem fazer. Estas são as possibilidades: ignorar a opção, descartar o pacote, descartar o pacote e enviar de volta um pacote ICMP e, ainda, a mesma opção anterior sem que, no entanto, sejam enviados pacotes ICMP para endereços multicasting (para impedir que um pacote de multicasting defeituoso gere milhões de relatórios ICMP).

Tamanho também é um campo de 1 byte. Ele identifica o tamanho do *Valor* (0 a 255 bytes), o qual contém todas as informações obrigatórias, com no máximo 255 bytes.

O cabeçalho hop-by-hop é usado para as informações que todos os roteadores ao longo do caminho devem examinar. Até agora, uma opção foi definida: compatibilidade com datagramas além de 64 KB. O formato desse cabeçalho é mostrado na Figura 5.50. Quando ele é usado, o campo *Tamanho da carga útil* do cabeçalho fixo é definido como zero.

A exemplo de todos os outros cabeçalhos de extensão, esse começa com 1 byte, cuja função é identificar o tipo de cabeçalho que vem a seguir. Depois desse byte, há um deles cuja função é identificar o tamanho do cabeçalho hop-by-hop, excluindo os primeiros 8 bytes, os quais são obrigatórios. Todas as extensões começam dessa maneira.

Os 2 bytes seguintes indicam que essa opção define o tamanho do datagrama (código 194) e que o tamanho é um número de 4 bytes. Os quatro últimos bytes mostram o tamanho do datagrama. Não são permitidos datagramas com menos de 65.536 bytes; datagramas maiores resultarão na eliminação do pacote no primeiro roteador e no envio de uma mensagem de erro ICMP. Os datagramas que utilizam essa extensão de cabeçalho são chamados **jumbogramas**. O uso de jumbogramas é importante para as aplicações de supercomputadores, que devem transferir gigabytes de dados pela Internet com eficiência.

Próximo cabeçalho	0	194	4
Tamanho da carga útil jumbo			

Figura 5.50 | O cabeçalho de extensão hop-by-hop para datagramas grandes (jumbogramas).

O cabeçalho de opções de destino é usado em campos que só precisam ser interpretados no host de destino. Na versão inicial do IPv6, as únicas opções definidas são opções nulas para preencher esse cabeçalho até formar um múltiplo de 8 bytes; portanto, inicialmente ele não será utilizado. Esse campo foi incluído para garantir que o novo software de roteamento e de host poderá tratá-lo, no caso de alguém imaginar uma opção de destino algum dia.

O cabeçalho de roteamento lista um ou mais roteadores que devem ser visitados no caminho até o destino. Ele é muito semelhante ao roteamento de origem livre do IPv4, no fato de que todos os endereços listados têm de ser visitados em ordem; porém, outros roteadores não listados também podem ser visitados. O formato do cabeçalho de roteamento é mostrado na Figura 5.51.

Os quatro primeiros bytes do cabeçalho de extensão para roteamento contêm quatro inteiros de 1 byte. Os campos *Próximo cabeçalho* e *Tamanho da extensão do cabeçalho* já foram descritos. O campo *Tipo de roteamento* fornece o formato do restante do cabeçalho. O tipo 0 informa que uma palavra reservada de 32 bits segue a primeira palavra, e é acompanhada por algum número de endereços IPv6. Outros tipos podem ser criados no futuro, se necessário. Por fim, o campo *Segmentos restantes* controla quantos endereços da lista ainda não foram visitados. Ele é decrementado toda vez que um endereço é visitado. Quando chega a 0, o pacote fica por sua própria conta, sem nenhuma orientação adicional sobre qual rota seguir. Em geral, a essa altura ele está tão perto do destino que a melhor rota é evidente.

O cabeçalho de fragmento lida com a fragmentação da mesma maneira que o IPv4. O cabeçalho contém o identificador do datagrama, o número do fragmento e um bit que informa se haverá novos fragmentos em seguida. No IPv6, ao contrário do IPv4, apenas o host de origem pode

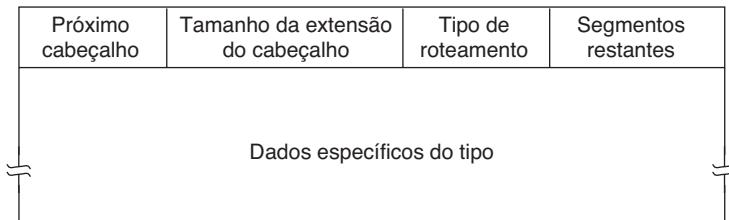



Figura 5.51 | O cabeçalho de extensão para roteamento.

fragmentar um pacote. Os roteadores ao longo do caminho não podem fazê-lo. Essa mudança é uma grande ruptura filosófica com o IP original, mas acompanha a prática atual para o IPv4. Além disso, ela simplifica o trabalho dos roteadores e faz o roteamento ir mais rápido. Como já dissemos, se um roteador for confrontado com um pacote muito grande, ele o descartará e enviará um pacote ICMP de volta à origem. Tais informações permitem que o host de origem utilize esse cabeçalho para fragmentar o pacote em pedaços menores e tentar outra vez.

O cabeçalho de autenticação oferece um mecanismo pelo qual o receptor de um pacote pode ter certeza de quem o enviou. A carga útil de segurança criptografada torna possível criptografar o conteúdo de um pacote, de modo que apenas o destinatário pretendido possa ler seu conteúdo. Esses cabeçalhos utilizam técnicas criptográficas, que descreveremos no Capítulo 8, para alcançar os objetivos a que se propõem.

Controvérsias

Considerando-se o processo de projeto aberto e o ardor com que as pessoas defenderam suas opiniões, não foi surpresa que muitas das escolhas feitas para o IPv6 tenham gerado, digamos assim, tanta polêmica. Vamos apresentar um breve resumo dessas controvérsias. Se desejar conhecer todos os detalhes, consulte as RFCs relevantes.

Já mencionamos o argumento sobre o tamanho do endereço. O resultado, fruto de uma conciliação, foi o uso de endereços de comprimento fixo de 16 bytes.

O tamanho do campo *Limite de hops* também provocou muita discussão. De um lado estavam os defensores da tese de que seria um grande equívoco limitar o número de hops a um máximo de 255 (implícito na utilização de um campo de 8 bits). Afinal de contas, os caminhos de 32 hops são comuns agora e, daqui a 10 anos, talvez sejam comuns caminhos muito mais longos. Essas pessoas argumentaram que a utilização de um tamanho de endereço gigantesco era algo inovador, mas usar um pequeno número de hops era retrógrado. Para elas, o maior pecado que a ciência da computação poderia cometer seria oferecer tão poucos bits.

Do outro lado estavam os que acreditavam que a ampliação excessiva do campo incharia o cabeçalho. Além disso, a função do campo *Limite de hops* é impedir que os pacotes vagueiem por muito tempo, tese incompatível com o tempo necessário para percorrer 65.535 hops. Por fim, com o crescimento da Internet, serão criados cada vez mais enlaces de longa distância, tornando possível ir de qualquer país a qualquer outro, usando no máximo meia dúzia de hops. Se forem necessários mais de 125 hops para ir da origem até o destino por seus respectivos gateways internacionais, algo estará errado com os backbones nacionais. Os defensores dos 8 bits venceram essa batalha.

Outro problema foi o tamanho máximo do pacote. A comunidade dos supercomputadores desejava pacotes com mais de 64 KB. Quando inicia a transferência, um supercomputador está tentando executar uma tarefa importantíssima e não deve ser interrompido a cada 64 KB. O argumento contrário ao uso de grandes pacotes é que, se um pacote de 1 MB alcançar uma linha T1 de 1,5 Mbps, ele a ocupará durante 5 segundos, o que produzirá um atraso bastante perceptível para os usuários interativos que estão compartilhando a linha. Chegou-se a um acordo quanto a essa questão: os pacotes normais foram limitados a 64 KB, mas o cabeçalho de extensão hop-by-hop pode ser usado para permitir a utilização de jumbogramas.

Outro assunto polêmico foi a remoção do checksum do IPv4. Algumas pessoas comparavam esse movimento à remoção dos freios de um automóvel. Dessa forma, o veículo ficaria mais leve e mais veloz, mas, se ocorresse alguma situação inesperada, haveria um sério problema.

O argumento contrário aos checksums levava em conta que qualquer aplicação que realmente se preocupasse com a integridade dos dados teria de incluir um checksum na camada de transporte e, por essa razão, seria uma redundância ter um novo checksum no IP (além do da camada de enlace de dados). Vale lembrar também que a experiência mostrava que o cálculo do checksum do IP representava um grande desperdício no IPv4. Os que eram contrários ao checksum venceram essa batalha, e o IPv6 ficou sem ele.

Os hosts móveis também foram motivo de discórdia. Quando um computador portátil vai de um canto a outro do mundo, ele pode continuar a operar no destino com o mesmo endereço IPv6 ou tem de usar um esquema com agentes nacionais e internacionais? Algumas pessoas quiseram dar ao IPv6 uma compatibilidade explícita com hosts móveis. Esse esforço fracassou, pois não foi possível chegar a um consenso quanto a essa questão.

Provavelmente a maior batalha se deu no campo da segurança. Todos concordavam que ela era essencial. O

problema foi descobrir onde e como usá-la. Primeiro onde. O argumento para inseri-la na camada de rede é que nessa camada ela se torna um serviço-padrão que todas as aplicações podem usar sem nenhum planejamento prévio. O argumento contrário é que, em geral, tudo o que as aplicações realmente seguras desejam é a criptografia, na qual a aplicação de origem cuida da codificação, e a aplicação de destino a desfaz. Se não for assim, o usuário estará à mercê de implementações potencialmente problemáticas, sobre as quais ele não tem o menor controle, na camada de rede. A resposta a esse argumento é que essas aplicações podem se abster de usar os recursos de segurança do IP, executando elas mesmas essa tarefa. A réplica a esse argumento é que as pessoas que não confiam na execução adequada dessa tarefa não estão dispostas a pagar o preço das desajeitadas e lentas implementações IP que dispõem desse recurso, ainda que ele esteja desativado.

Outro aspecto a ser levado em consideração quanto à localização da segurança diz respeito ao fato de que, em muitos países (mas não em todos), as leis de exportação que envolvem a criptografia são muito rígidas. Alguns deles, como a França e o Iraque, também impõem inúmeras restrições quanto a seu uso doméstico; portanto, as pessoas não podem guardar segredos do governo. Consequentemente, os Estados Unidos (e muitos outros países) não teriam mercado consumidor para nenhuma implementação do IP que usasse um sistema criptográfico suficientemente forte para ser digno de valor. A maioria dos fabricantes de computadores detesta produzir dois conjuntos de software, um para uso doméstico e outro para exportação.

Um ponto em que não houve controvérsia é que ninguém espera que a Internet usando IPv4 seja desligada em um domingo à noite e volte como uma Internet usando IPv6 na manhã de segunda-feira. Em vez disso, 'ilhas' isoladas de IPv6 serão convertidas, inicialmente comunicando-se por meio de túneis, conforme mostramos na Seção 5.5.3. À medida que as ilhas IPv6 crescerem, elas se fundirão formando ilhas maiores. Por fim, todas as ilhas se fundirão e a Internet estará totalmente convertida.

Pelo menos esse foi o plano. A implementação provou ser o calcanhar de aquiles do IPv6. Ele continua sendo pouco usado, embora todos os principais sistemas operacionais o admitam totalmente. A maioria das implementações é de situações novas, em que um operador de rede — por exemplo, um operador de telefonia móvel — precisa de um grande número de endereços IP. Muitas estratégias têm sido definidas para ajudar a facilitar a transição. Entre elas estão maneiras de configurar automaticamente os túneis que transportam o IPv6 pela Internet IPv4, e maneiras para os hosts encontrarem automaticamente as extremidades do túnel. Os hosts de pilha dupla têm uma implementação IPv4 e IPv6, de modo que possam selecionar qual protocolo utilizar dependendo do destino do pacote. Essas estratégias facilitarão a implantação substancial que parece inevitável quando os endereços IPv4 se esgotarem. Para obter mais informações sobre o IPv6, consulte Davies (2008).

5.6.4 Protocolos de controle da Internet

Além do IP, que é usado para transferência de dados, a Internet tem diversos protocolos de controle que são usados na camada de rede. Eles incluem ICMP, ARP e DHCP. Nesta seção, veremos cada um deles por vez, descrevendo as versões que correspondem ao IPv4, pois são protocolos que estão em uso. ICMP e DHCP possuem versões semelhantes para o IPv6; o equivalente do ARP é chamado NDP (Neighbor Discovery Protocol) para o IPv6.

ICMP – Internet Control Message Protocol

A operação da Internet é monitorada de perto pelos roteadores. Quando acontece algo inesperado durante o processamento do pacote em um roteador, o evento é relatado ao transmissor pelo **protocolo de mensagem de controle da Internet**, ou **ICMP** (**Internet Control Message Protocol**). O ICMP também é usado para testar a Internet. Cerca de 12 tipos de mensagens ICMP são definidos. Cada tipo de mensagem ICMP é transportada encapsulada dentro de um pacote IP. Os mais importantes estão listados na Tabela 5.8.

A mensagem DESTINATION UNREACHABLE é usada quando o roteador não pode localizar o destino ou quando um pacote com o bit *DF* não pode ser entregue porque uma rede com 'pacote pequeno' se encontra no caminho.

A mensagem TIME EXCEEDED é enviada quando um pacote é descartado, pois seu contador de *TTL* (*time to live*) alcançou zero. Esse evento é um sintoma de que os pacotes estão realizando um looping ou que os valores do contador estão sendo definidos com um valor muito baixo.

Um uso inteligente dessa mensagem de erro é o utilitário **traceroute**, que foi desenvolvido por Van Jacobson em 1987. O traceroute encontra os roteadores ao longo do caminho do host para um endereço IP de destino. Ele encontra essa informação sem nenhum tipo de suporte de rede privilegiado. O método é simplesmente enviar uma sequência de pacotes para o destino, primeiro com um TTL de 1, depois um TTL de 2, 3 e assim por diante. Os contadores nesses pacotes alcançarão zero em roteadores sucessivos ao longo do caminho. Cada um desses roteadores enviará fielmente uma mensagem TIME EXCEEDED de volta ao host. Por essas mensagens, o host pode determinar os endereços IP dos roteadores ao longo do caminho, além de manter estatísticas e tempos sobre as partes do caminho. Não foi para isso que a mensagem TIME EXCEEDED foi criada, mas talvez ela seja a ferramenta de depuração mais poderosa de todos os tempos.

A mensagem PARAMETER PROBLEM indica que um valor ilegal foi detectado no campo de cabeçalho. Esse problema indica um bug no software de rede do IP do host transmissor ou possivelmente no software de rede do roteador no caminho.

A mensagem SOURCE QUENCH há muito tempo era usada para restringir os hosts que estavam enviando pacotes em demasia. Quando um host recebia essa mensagem, ele deveria desacelerar. Ela raramente continua a ser usada

Tipo de mensagem	Descrição
Destination unreachable	O pacote não pôde ser entregue
Time exceeded	O campo TTL atingiu 0
Parameter problem	Campo de cabeçalho inválido
Source quench	Restringe o envio de pacotes
Redirect	Ensina uma rota a um roteador
Echo e Echo reply	Verificam se uma máquina está ativa
Timestamp request/reply	O mesmo que Echo, mas com registro de tempo
Router advertisement/solicitation	Encontra um roteador próximo

Tabela 5.8 | Os principais tipos de mensagem ICMP.

porque, quando ocorre congestionamento, esses pacotes tendem a acrescentar mais combustível ao fogo, e não é correto responder a eles. O controle de congestionamento na Internet agora é feito em grande parte tomando ação na camada de transporte, usando perdas de pacotes como um sinal de congestionamento; vamos estudá-lo em detalhes no Capítulo 6.

A mensagem REDIRECT é usada quando um roteador observa que um pacote parece estar roteado incorretamente. Ela é usada pelo roteador dizendo ao host transmissor para atualizar para uma rota melhor.

As mensagens ECHO e ECHO REPLY são enviadas pelos hosts para ver se determinado destino pode ser alcançado e está atualmente ativo. Ao receber a mensagem ECHO, o destino deve enviar de volta uma mensagem ECHO REPLY. Essas mensagens são usadas no utilitário **ping**, que verifica se um host está ativo na Internet.

As mensagens TIMESTAMP REQUEST e TIMESTAMP REPLY são semelhantes, exceto que o tempo de chegada da mensagem e o tempo de saída da resposta são registrados na resposta. Esse serviço pode ser usado para medir o desempenho da rede.

As mensagens ROUTER ADVERTISEMENT e ROUTER SOLICITATION são usadas para permitir que os hosts encontrem roteadores vizinhos. Um host precisa descobrir o endereço IP de pelo menos um roteador para poder enviar pacotes pela rede local.

Além dessas mensagens, outras foram definidas. A lista on-line agora é mantida em www.iana.org/assignments/icmp-parameters.

ARP — Address Resolution Protocol

Embora cada máquina na Internet tenha um ou mais endereços IP, estes não são suficientes para enviar pacotes. As placas de interface de rede, ou NICs (Network Interface Cards) da camada de enlace de dados, como as placas Ethernet, não entendem endereços da Internet. No caso da Ethernet, cada NIC já fabricado vem equipado com um endereço Ethernet exclusivo de 48 bits. Os fabricantes de NICs Ethernet solicitam um bloco de endereços Ethernet ao IEEE para garantir que duas NICs não tenham o mesmo endereço (para evitar conflitos caso duas NICs apareçam na mesma LAN). As NICs enviam e recebem quadros com base nos endereços Ethernet de 48 bits. Elas não conhecem nada a respeito de endereços IP de 32 bits.

A questão agora é: como os endereços IP são mapeados em endereços da camada de enlace de dados, como a Ethernet? Para explicar como isso funciona, vamos usar o exemplo da Figura 5.52, em que é ilustrada uma pequena universidade com redes /24. Uma rede (CC) é uma Ethernet comutada no departamento de Ciência da Computação. Ela tem o prefixo 192.32.65.0/24. A outra LAN (EE), também Ethernet comutada, está no departamento de Engenharia Elétrica e tem o prefixo 192.32.63.0/24. As duas LANs são conectadas por um roteador IP. Cada máquina em uma Ethernet e cada interface no roteador têm um endereço Ethernet único, rotulado de *E1* a *E6*, e um endereço IP único na rede CC ou EE.

Vamos começar vendo como um usuário no host 1 envia um pacote para um usuário no host 2 na rede CC. Vamos supor que o transmissor conheça o nome do receptor pretendido, possivelmente algo como *eagle.cs.uni.edu*. O primeiro passo é encontrar o endereço IP para o host 2. Essa pesquisa é realizada pelo DNS, o qual estudaremos no Capítulo 7. Por enquanto, vamos considerar apenas que o DNS retorna o endereço IP para o host 2 (192.32.65.5).

O software da camada superior no host 1 agora monta um pacote com 192.32.65.5 no campo *Endereço de destino* e passa para o IP transmitir. O IP pode examinar o endereço e ver que o destino está na rede CC (ou seja, sua própria rede). Porém, ele ainda precisa, de alguma forma, encontrar o endereço Ethernet do destino para enviar o quadro. Uma solução é ter um arquivo de configuração em algum lugar no sistema que mapeie os endereços IP em endereços Ethernet. Embora essa solução certamente seja possí-

vel, para organizações com milhares de máquinas, manter todos esses arquivos atualizados é uma tarefa demorada e passível de erros.

Uma solução melhor é que o host 1 envie um pacote de broadcast para a Ethernet perguntando quem possui o endereço IP 192.32.65.5. O broadcast chegará a cada máquina na Ethernet CC, e cada uma verificará seu endereço IP. O host 2 sozinho responderá com seu endereço Ethernet (*E2*). Desse modo, o host 1 aprenderá que o endereço IP 192.32.65.5 está no host com o endereço Ethernet *E2*. O protocolo usado para fazer essa pergunta e obter uma resposta é chamado **ARP** (**Address Resolution Protocol**). Quase toda máquina na Internet o utiliza. O ARP é definido na RFC 826.

A vantagem de usar o ARP sobre os arquivos de configuração é a simplicidade. O gerenciador do sistema não precisa fazer muito, exceto atribuir a cada máquina um endereço IP e decidir sobre as máscaras de sub-rede. O ARP faz o restante.

Nesse ponto, o IP no host 1 monta um quadro Ethernet endereçado a *E2*, põe o pacote IP (endereçado a 192.32.65.5) no campo de carga útil e o coloca na Ethernet. Os endereços IP e Ethernet desse pacote são mostrados na Figura 5.52. O NIC Ethernet do host 2 detecta esse quadro, reconhece-o como um quadro por si só e causa uma interrupção. O driver Ethernet extrai o pacote IP da carga útil e o passa para o IP, que vê que ele está endereçado corretamente e o processa.

Diversas otimizações são possíveis para fazer o ARP funcionar de modo mais eficiente. Para começar, quando uma máquina executa o ARP, ela coloca o resultado em cache caso precise entrar em contato com a mesma máquina em breve. Da próxima vez, ela encontrará o mapeamento em seu próprio cache, eliminando, assim, a necessidade de um segundo broadcast. Em muitos casos, o host 2 precisará enviar uma resposta de volta, o que o força também a executar o ARP para determinar o endereço Ethernet do transmissor. Esse broadcast ARP pode ser movido fazendo com que o host 1 inclua seu mapeamento IP para Ethernet no pacote ARP. Quando o broadcast ARP chega ao host 2, o par (192.32.65.7, *E1*) é inserido no cache ARP do host 2. Na verdade, todas as máquinas na Ethernet podem entrar com esse mapeamento em seus caches ARP.

Para permitir que os mapeamentos mudem, por exemplo, quando um host é configurado para usar um novo endereço IP (mas manter seu endereço Ethernet antigo), as entradas no cache ARP devem esgotar seu tempo-limite após alguns minutos. Um modo mais inteligente de ajudar a manter atualizada a informação em cache e otimizar o desempenho é fazer com que cada máquina envie seu mapeamento por broadcast quando for configurada. Esse broadcast geralmente é feito na forma de um ARP procurando seu próprio endereço IP. Não deve haver uma resposta, mas um efeito colateral do broadcast é criar ou atualizar uma entrada no cache ARP de cada máquina. Isso é conhecido como **ARP gratuito**. Se uma resposta chegar (inesperadamente), duas máquinas receberam o mesmo endereço IP. O erro deve ser resolvido pelo administrador da rede antes que as duas máquinas possam usá-lo.

Agora vejamos a Figura 5.52 novamente, mas desta vez considere que o host 1 deseja enviar um pacote para o host 4 (192.32.63.8) na rede EE. O host 1 verá que o endereço IP de destino não está na rede CC. Ele sabe enviar todo esse tráfego fora da rede para o roteador, que também é conhecido como o **gateway-padrão**. Por convenção, o gateway-padrão é o endereço mais baixo na rede (198.31.65.1). Para enviar um quadro ao roteador, o host 1 ainda deve conhecer o endereço Ethernet da interface do roteador na rede CC. Ele descobre isso enviando um broadcast ARP para 198.31.65.1, do qual descobre *E3*. Depois, ele envia o quadro. Os mesmos mecanismos de pesquisa são usados para enviar um pacote de um roteador para o

Quadro	IP de origem	Ethernet de origem	IP de destino	Ethernet de destino
Host 1 para 2, na rede CC	IP1	E1	IP2	E2
Host 1 para 4, na rede CC	IP1	E1	IP4	E3
Host 1 para 4, na rede EE	IP1	E4	IP4	E6

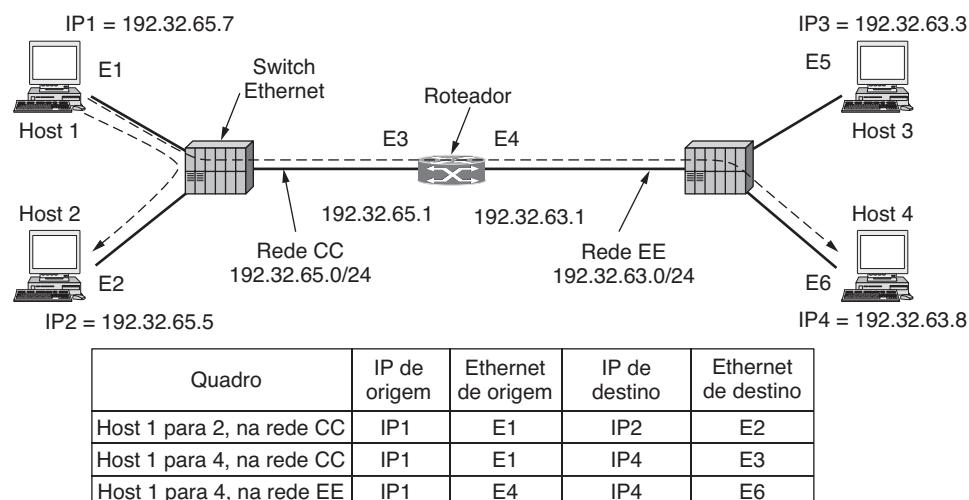



Figura 5.52 | Duas LANs Ethernet comutadas, unidas por um roteador.

seguinte, por uma sequência de roteadores em um caminho na Internet.

Quando a NIC Ethernet do roteador recebe esse quadro, ela entrega o pacote ao IP. Ela sabe pelas máscaras de rede que o pacote deve ser enviado para a rede EE, onde alcançará o host 4. Se o roteador não souber o endereço Ethernet para o host 4, então ele usará o ARP novamente. A tabela na Figura 5.52 lista os endereços Ethernet e IP de origem e destino que estão presentes nos quadros, conforme observado nas redes CC e EE. Observe que os endereços Ethernet mudam com o quadro em cada rede, enquanto os endereços IP permanecem constantes (pois indicam as extremidades através de todas as redes interconectadas).

Também é possível enviar um pacote do host 1 ao host 4 sem que o host 1 saiba que o host 4 está em uma rede diferente. A solução é fazer com que o roteador responda aos ARPs na rede CC para o host 4 e dê seu endereço Ethernet, *E3*, como resposta. Não é possível fazer com que o host 4 responda diretamente, pois ele não verá a solicitação ARP (pois os roteadores não encaminha broadcasts em nível de Ethernet). O roteador, então, recebe quadros enviados para 192.32.63.8 e os encaminha para a rede EE. Essa solução é chamada **proxy ARP**. Ela é usada em casos especiais, em que um host deseja aparecer em uma rede, embora na verdade resida em outra. Uma situação comum, por exemplo, é um computador móvel que deseja que algum outro nó capture pacotes para ele quando não estiver em sua rede doméstica.

DHCP — Dynamic Host Configuration Protocol

O ARP (bem como outros protocolos da Internet) pressupõe que os hosts são configurados com alguma informação básica, como seus próprios endereços IP. Como os hosts obtêm essa informação? É possível configurar manualmente cada computador, mas isso é tedioso e passível de erros. Existe um modo melhor, chamado protocolo de configuração dinâmica de host, ou **DHCP** (**Dynamic Host Configuration Protocol**).

Com DHCP, cada rede precisa ter um servidor DHCP responsável pela configuração. Quando um computador é iniciado, ele tem um endereço Ethernet ou outro endereço da camada de enlace embutido na NIC, mas não um endereço IP. Assim como o ARP, o computador envia uma solicitação de broadcast por endereço IP em sua rede. Ele faz isso usando um pacote DHCP DISCOVER. Esse pacote precisa alcançar o servidor DHCP. Se esse servidor não estiver conectado diretamente à rede, o roteador será configurado para receber broadcasts DHCP e repassá-los ao servidor DHCP, onde quer que esteja localizado.

Quando o servidor recebe a solicitação, ele aloca um endereço IP livre e o envia ao host em um pacote DHCP OFFER (que novamente pode ser repassado pelo roteador). Para poder fazer isso funcionar até mesmo quando os hosts não têm endereços IP, o servidor identifica um host usando seu endereço Ethernet (que é transportado no pacote DHCP DISCOVER).

Um problema que surge com a atribuição automática de endereços IP a partir de um pool é por quanto tempo um endereço IP deve ser alocado. Se um host sair da rede e não retornar seu endereço IP ao servidor DHCP, esse endereço será permanentemente perdido. Após certo tempo, muitos endereços podem se perder. Para impedir que isso aconteça, a atribuição de endereço IP pode ser por um período de tempo fixo, em uma técnica chamada **leasing**. Imediatamente antes que o tempo de validade de leasing termine, o host precisa pedir uma renovação ao DHCP. Se ele deixar de fazer uma solicitação ou a solicitação for negada, o host não pode mais usar o endereço IP que lhe foi dado anteriormente.

O DHCP é descrito nas RFCs 2131 e 2132. Ele é muito usado na Internet para configurar todos os tipos de parâmetros, além de oferecer endereços IP aos hosts. Tanto em redes empresariais como domésticas, o DHCP é usado pelos ISPs para definir os parâmetros dos dispositivos pelo enlace de acesso à Internet, de modo que os clientes não precisam ligar para seus ISPs para receber essa informação. Alguns exemplos comuns da informação configurada incluem a máscara de rede, o endereço IP do gateway-padrão e os endereços IP de DNS e servidores de hora. O DHCP em grande parte substituiu os protocolos anteriores (chamados RARP e BOOTP) com funcionalidades bem mais limitadas.

5.6.5 Rótulos de comutação e MPLS

Até aqui, em nosso passeio pela camada de rede da Internet, focalizamos exclusivamente os pacotes como datagramas encaminhados pelos roteadores IP. Também há outro tipo de tecnologia que está começando a ser largamente empregada, especialmente pelos ISPs, a fim de mover o tráfego da Internet por suas redes. Essa tecnologia se chama **MPLS** (**MultiProtocol Label Switching**) e está perigosamente perto da comutação de circuitos. Apesar do fato de muitas pessoas na comunidade da Internet terem uma grande aversão pelas redes orientadas à conexão, a ideia parece estar voltando. Como Yogi Berra disse certa vez, isso é como um *déjà vu* de tudo novamente. Porém, existem diferenças essenciais entre o modo como a Internet lida com a construção da rota e a maneira como as redes orientadas à conexão fazem isso, de forma que a técnica certamente não é a comutação de circuitos tradicional.

O MPLS acrescenta um rótulo na frente de cada pacote, e o encaminhamento é baseado no rótulo, em vez do endereço de destino. Transformar o rótulo em um índice para uma tabela interna faz da descoberta da interface de saída correta simplesmente uma questão de pesquisa de tabela. Usando essa técnica, o encaminhamento pode ser feito muito rapidamente. Essa vantagem foi a motivação original por trás do MPLS, que começou como uma tecnologia patenteada, conhecida por vários nomes, incluindo

comutação de tags. Por fim, a IETF começou a padronizar a ideia. Ela é descrita na RFC 3031 e em muitas outras RFCs. Os principais benefícios com o tempo foram o roteamento que é flexível e o encaminhamento que é adequado para a qualidade de serviço, assim como é rápido.

A primeira pergunta a fazer é: onde entra o rótulo? Como os pacotes IP não foram projetados para circuitos virtuais, não existe um campo disponível para os números de circuito virtual dentro do cabeçalho IP. Por esse motivo, um novo cabeçalho MPLS teve de ser acrescentado na frente do cabeçalho IP. Em uma conexão roteador a roteador usando PPP como o protocolo de enquadramento, o formato do quadro, incluindo os cabeçalhos PPP, MPLS, IP e TCP, pode ser visto na Figura 5.53.

O cabeçalho MPLS genérico tem 4 bytes de extensão e quatro campos. O mais importante é o campo *Rótulo*, que mantém o índice. O campo *QoS* indica a classe de serviço. O campo *S* relaciona-se ao empilhamento de vários rótulos (que discutiremos mais adiante). O campo *TTL* indica quantas outras vezes mais o pacote pode ser encaminhado. Ele é decrementado em cada roteador e, se atingir 0, é descartado. Esse recurso impede o loop infinito no caso da instabilidade do roteamento.

O MPLS fica entre o protocolo da camada de rede IP e o protocolo da camada de enlace PPP. Ela não é realmente um protocolo da camada 3, pois depende do IP ou de outros endereços da camada de rede para estabelecer caminhos por rótulo. Ela também não é realmente um protocolo da camada 2, pois encaminha pacotes por vários hops, e não por um único enlace. Por esse motivo, o MPLS às vezes é descrito como um protocolo da camada 2,5. Ela é uma ilustração de que os protocolos reais nem sempre se encaixam bem em nosso modelo de protocolos em camadas.

No lado positivo, como os cabeçalhos MPLS não fazem parte do pacote da camada de rede ou do quadro da camada de enlace de dados, o MPLS é, em grande parte, independente das duas camadas. Entre outras coisas, essa propriedade significa que é possível montar switches MPLS que possam encaminhar tanto pacotes IP quanto pacotes que não sejam IP, dependendo do que aparecer. É desse recurso que vem o 'multiprotocolo' no nome MPLS. O MPLS também pode transportar pacotes IP por redes não IP.

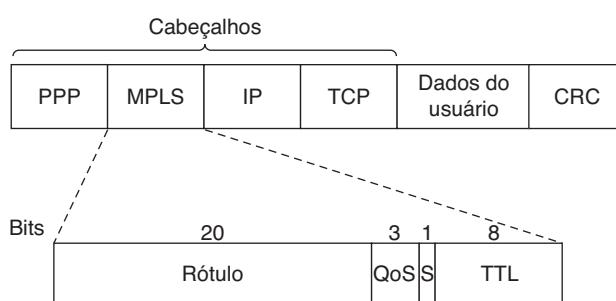



Figura 5.53 | Transmitindo um segmento TCP usando IP, MPLS e PPP.

Quando um pacote melhorado com MPLS chega a um **LSR** (**Label Switched Router**), o rótulo é usado como um índice para uma tabela, a fim de determinar a interface de saída e também o novo rótulo a usar. Essa comutação de rótulos é usada em todas as redes de circuito virtual. Os rótulos têm significado apenas local e dois roteadores podem alimentar pacotes não relacionados com o mesmo rótulo em outro roteador para transmissão pela mesma interface de saída. Para que sejam distinguidos na outra ponta, os rótulos devem ser remapeados em cada hop. Vimos esse mecanismo em ação na Figura 5.3. O MPLS usa a mesma técnica.

A propósito, algumas pessoas distinguem entre *encaminhamento* e *comutação*. O encaminhamento é o processo de encontrar uma melhor combinação para um endereço de destino em uma tabela, para decidir para onde enviar pacotes. Um exemplo é o algoritmo do maior prefixo combinado, usado para o encaminhamento IP. Ao contrário, a comutação usa um rótulo tirado do pacote como um índice para uma tabela de encaminhamento. Isso é mais simples e mais rápido. Essas definições, porém, estão longe de ser unânimes.

Como a maioria dos hosts e roteadores não entende o MPLS, também devemos perguntar quando e como os rótulos são anexados aos pacotes. Isso acontece quando um pacote IP alcança a borda de uma rede MPLS. O **roteador de borda de rótulo**, ou **LER** (**Label Edge Router**), inspeciona o endereço IP de destino e outros campos para ver qual caminho na rede MPLS o pacote deve seguir, e coloca o rótulo correto na frente do pacote. Dentro da rede MPLS, esse rótulo é usado para encaminhar o pacote. Na outra borda da rede MPLS, o rótulo já terá atendido à sua finalidade e será removido, revelando novamente o pacote IP para a próxima rede. Esse processo pode ser visto na Figura 5.54. Uma diferença dos circuitos virtuais tradicionais é o nível de agregação. Certamente, é possível que cada fluxo tenha seu próprio conjunto de rótulos pela rede MPLS. Porém, é mais comum que os roteadores agrupem vários fluxos que terminam em determinado roteador ou LAN e usem um único rótulo para eles. Diz-se que os fluxos agrupados sob um único rótulo pertencem à mesma **classe de equivalência de encaminhamento**, ou **FEC** (**Forwarding Equivalence Class**). Essa classe abrange não apenas aonde os pacotes estão indo, mas também sua classe de serviço (no sentido dos serviços diferenciados), pois todos os pacotes são tratados da mesma maneira para fins de encaminhamento.

Com o roteamento tradicional de circuito virtual, não é possível agrupar vários caminhos distintos com diferentes extremidades no mesmo identificador de circuito virtual, pois não haveria como distingui-los no destino. Com o MPLS, os pacotes ainda contêm seu endereço de destino, além do rótulo. Ao final da rota, o cabeçalho do rótulo pode ser removido e o encaminhamento pode continuar normalmente, usando o endereço de destino da camada de rede.

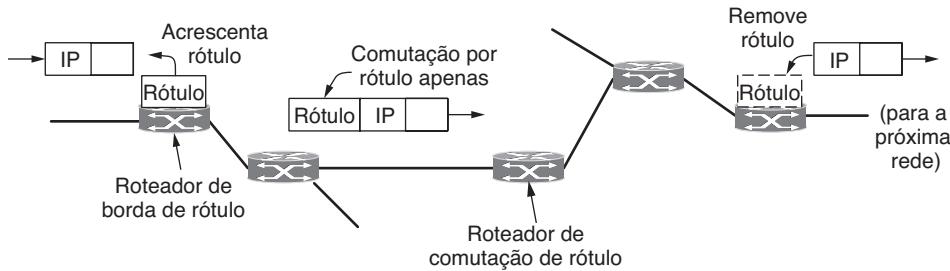



Figura 5.54 | Encaminhando um pacote IP por uma rede MPLS.

Na realidade, o MPLS vai ainda mais adiante. Ele pode operar em vários níveis ao mesmo tempo, acrescentando mais de um rótulo à frente de um pacote. Por exemplo, suponha que existam muitos pacotes que já tenham diferentes rótulos (pois queremos tratar os pacotes de forma diferente em algum lugar na rede) que deverão seguir um caminho comum até algum destino. Em vez de estabelecer muitos caminhos de comutação de rótulos, um para cada rótulo diferente, podemos estabelecer um único caminho. Quando os pacotes já rotulados atingem o início desse caminho, outro rótulo é acrescentado à frente. Isso é chamado de pilha de rótulos. O rótulo mais externo guia os pacotes ao longo do caminho. Ele é removido no final do caminho, os rótulos são revelados e, se houver alguns, são usados para encaminhar o pacote mais adiante. O bit *S* na Figura 5.53 permite que um roteador removendo um rótulo saiba se ainda existem rótulos adicionais. Ele é definido como 1 para o rótulo inferior e 0 para todos os outros rótulos.

A última pergunta que faremos é como as tabelas de encaminhamento de rótulos são montadas de modo que os pacotes as sigam. Essa é uma área de diferença importante entre MPLS e projetos convencionais de circuito virtual. Nas redes tradicionais de circuito virtual, quando um usuário quer estabelecer uma conexão, um pacote de configuração é disparado na rede, para criar o caminho e as entradas da tabela de encaminhamento. O MPLS não envolve os usuários na fase de configuração. Exigir que os usuários façam algo diferente de enviar um datagrama atrapalharia grande parte do software da Internet existente.

Em vez disso, a informação de encaminhamento é estabelecida por protocolos, que são uma combinação de protocolos de roteamento e de estabelecimento de conexão. Esses protocolos de controle são nitidamente separados do encaminhamento de rótulos, o que permite que vários protocolos de controle sejam utilizados. Uma das variantes funciona desta maneira: quando um roteador é iniciado, ele verifica quais rotas até o destino (por exemplo, quais prefixos pertencem às suas interfaces). Depois ele cria um ou mais FECs para elas, aloca um rótulo para cada uma e passa os rótulos a seus vizinhos. Estes, por sua vez, entram com os rótulos em suas tabelas de encaminhamento e enviam novos rótulos aos seus vizinhos, até que todos os roteadores tenham adquirido o caminho. Recursos também podem ser reservados à medida que o caminho é construído, para garantir uma qualidade de serviço apropriada. Outras variantes podem estabelecer caminhos diferentes, como caminhos de engenharia de tráfego que levam em conta a capacidade não usada, e criar caminhos por demanda para dar suporte a ofertas de serviço, como a qualidade.

Embora as ideias básicas por trás do MPLS sejam simples, os detalhes são complicados, com muitas variantes e casos de uso que estão sendo ativamente desenvolvidos. Para obter mais informações, consulte Davie e Farrel (2008) e Davie e Rekhter (2000).

5.6.6 OSPF — PROTOCOLO DE ROTEAMENTO DE GATEWAY INTERIOR

Neste ponto, já terminamos nosso estudo de como os pacotes são encaminhados na Internet. É hora de prosseguir para o próximo assunto: roteamento na Internet. Como já dissemos, a Internet é composta de um grande número de redes independentes, ou **sistemas autônomos** (**Autonomous Systems** — **AS**), que são operados por diferentes organizações, normalmente uma empresa, universidade ou ISP. Dentro de sua própria rede, uma organização pode usar seu próprio algoritmo para roteamento interno, ou **roteamento intradomínio**, como normalmente é mais conhecido. Apesar disso, existem apenas alguns protocolos-padrão que são populares. Nesta seção, estudaremos o problema de roteamento intradomínio e veremos o protocolo OSPF, que é bastante utilizado na prática. Um protocolo de roteamento intradomínio também é chamado **protocolo gateway interior**. Na próxima seção, estudaremos o problema de roteamento entre redes operadas independentemente, ou **roteamento interdomínio**. Para esse caso, todas as redes devem usar o mesmo protocolo de roteamento interdomínio ou **protocolo de gateway exterior**. O protocolo usado na Internet é o BGP (Border Gateway Protocol).

Os primeiros protocolos de roteamento intradomínio usavam um algoritmo por vetor de distância, baseado no algoritmo de Bellman-Ford distribuído, herdado da ARPANET. O RIP (Routing Information Protocol) é o prin-

cipal exemplo que é usado até os dias de hoje. Ele funciona bem em sistemas pequenos, mas não muito bem quando as redes se tornam maiores. Ele também sofre do problema de contagem ao infinito e de convergência geralmente lenta. A ARPANET passou para um protocolo de estado de enlace em maio de 1979, em decorrência desses problemas, e em 1988 a IETF começou a trabalhar em um protocolo de estado de enlace para o roteamento intradomínio. Esse protocolo, chamado **OSPF** (**Open Shortest Path First**), tornou-se padrão em 1990. Ele ocasionou um protocolo chamado **IS-IS** (**Intermediate-System to Intermediate-System**), que se tornou um padrão ISO. Em virtude de sua herança compartilhada, os dois protocolos são muito mais semelhantes do que diferentes. Para ver a história completa, consulte a RFC 2328. Eles são os protocolos de roteamento intradomínio mais difundidos, e a maioria dos fornecedores de roteador agora oferece suporte para ambos. O OSPF é mais utilizado em redes de empresas, e o IS-IS é mais usado em redes ISP. Faremos um esboço de como o OSPF funciona.

Dada a longa experiência com outros protocolos de roteamento, o grupo que estava projetando o OSPF tinha uma longa lista de requisitos que precisavam de ser atendidos. Primeiro, o algoritmo tinha de ser publicado na literatura aberta, daí o 'O' (Open) em OSPF. Uma solução patenteada, pertencente a uma empresa, não serviria. Em segundo lugar, o novo protocolo deveria dar suporte a uma série de métricas de distância, incluindo distância física, atraso e assim por diante. Em terceiro lugar, ele tinha de ser um algoritmo dinâmico, que se adaptasse às mudanças na topologia de maneira automática e rápida.

Em quarto lugar, e novo para o OSPF, ele tinha de dar suporte ao roteamento com base no tipo de serviço. O novo protocolo deveria ser capaz de rotear o tráfego em tempo real de uma maneira e o restante do tráfego de outra. Na época, o IP tinha um campo *Tipo de serviço*, mas nenhum protocolo de roteamento existente o utilizava. Esse campo foi incluído no OSPF, mas ninguém o usava ainda, e, por fim, foi removido. Talvez esse requisito tenha estado à frente de seu tempo, e precedeu o trabalho da IETF sobre serviços diferenciados, que rejuvenesceu as classes de serviço.

Em quinto lugar, e relacionado ao anterior, o OSPF tinha de realizar balanceamento de carga, dividindo-a por várias conexões. A maioria dos protocolos anteriores enviava todos os pacotes por uma única melhor rota, mesmo que houvesse duas rotas que fossem igualmente boas. A outra rota nem sequer era usada. Em muitos casos, a divisão da carga por várias rotas oferece melhor desempenho.

Em sexto lugar, o suporte para sistemas hierárquicos era necessário. Em 1988, algumas redes tinham se tornado tão grandes que não se poderia esperar que algum roteador conhecesse a topologia inteira. O OSPF tinha de ser projetado de modo que nenhum roteador precisasse disso.

Em sétimo lugar, algum modo de segurança era exigido para impedir que estudantes procurando diversão bisbilhotassem os roteadores enviando-lhes informações de roteamento falsas. Finalmente, era preciso algum meio de lidar com os roteadores que estavam conectados à Internet por meio de um túnel. Os protocolos anteriores não cuidavam disso muito bem.

O OSPF tem suporte para enlaces ponto a ponto (por exemplo, SONET) e redes de broadcast (por exemplo, a maioria das LANs). Na realidade, ele é capaz de dar suporte a redes com vários roteadores, cada um deles podendo se comunicar diretamente com os outros (chamadas **redes de acesso múltiplo**), mesmo que eles não tenham capacidade de broadcast. Os protocolos anteriores também não cuidavam desse caso muito bem.

Um exemplo de uma rede de sistema autônomo aparece na Figura 5.55(a). Os hosts são omitidos porque geralmente não desempenham um papel no OSPF, ao contrário dos roteadores e das redes (que podem conter hosts). A maioria dos roteadores na Figura 5.55(a) é conectada a outros roteadores por enlaces ponto a ponto e às redes para alcançar os hosts nessas redes. Porém, os roteadores *R3*, *R4* e *R5* são conectados por uma LAN de broadcast, como a Ethernet comutada.

O OSPF opera abstraindo a coleção de redes, roteadores e enlaces reais em um grafo direcionado no qual cada arco recebe um peso (distância, atraso etc.). Uma conexão ponto a ponto entre dois roteadores é representada por um par de arcos, um em cada direção. Seus pesos podem ser diferentes. Uma rede de broadcast é representada por um nó para a própria rede, mais um nó para cada roteador. Os arcos desse nó da rede para os roteadores têm peso 0. Apesar disso, eles são importantes, pois, sem eles, não existe caminho pela rede. Outras redes, que possuem apenas hosts, têm apenas um arco chegando até elas, e não um retornando. Essa estrutura gera rotas aos hosts, mas não através deles.

A Figura 5.55(b) mostra a representação gráfica da rede da Figura 5.55(a). O que o OSPF fundamentalmente faz é representar a rede real como um grafo como este e, depois, usar o método de estado de enlace para que cada roteador calcule o caminho mais curto de si mesmo para todos os outros nós. Múltiplos caminhos podem ser encontrados, os quais são igualmente curtos. Nesse caso, o OSPF se lembra do conjunto de caminhos mais curtos e, durante o encaminhamento de pacotes, o tráfego é divido entre eles. Isso ajuda a balancear a carga. Isso é chamado **ECMP** (**Equal Cost MultiPath**).

Muitos dos ASs na Internet são, por si sós, grandes e não triviais para administrar. Para trabalhar nessa escala, o OSPF permite que um AS seja dividido em **áreas** numeradas, onde uma área é uma rede ou um conjunto de redes contíguas. As áreas não se sobrepõem, mas não precisam ser completas, ou seja, alguns roteadores podem não

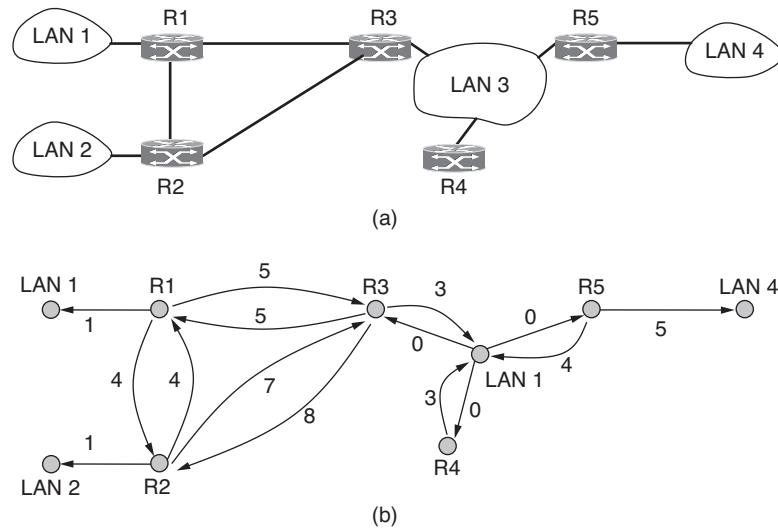



Figura 5.55 | (a) Um sistema autônomo. (b) Uma representação gráfica de (a).

pertencer a uma área. Os roteadores que se encontram totalmente dentro de uma área são chamados **roteadores internos**. Uma área é uma generalização de uma rede individual. Fora de uma área, seus destinos são visíveis, mas não sua topologia. Essa característica ajuda na expansão do roteamento.

Cada AS tem uma **área de backbone**, chamada área 0. Os roteadores nessa área são chamados **roteadores de backbone**. Todas as áreas são conectadas ao backbone, possivelmente por túneis, de modo que é possível seguir de qualquer área no AS para qualquer outra área nele por meio do backbone. Um túnel é representado no grafo simplesmente como outro arco com um custo. Assim como outras áreas, a topologia do backbone não é visível fora dele.

Cada roteador que está conectado a duas ou mais áreas é chamado **roteador de borda de área**. Ele também precisa fazer parte do backbone. A tarefa de um roteador de borda de área é resumir os destinos em uma área e injetar esse resumo nas outras às quais ele está conectado. Esse resumo inclui informação de custo, mas nem todos os detalhes da topologia dentro de uma área. A passagem de informações de custo permite que os hosts em outras áreas encontrem o melhor roteador de borda a usar para entrar em uma área. Não passar informação de topologia reduz o tráfego e simplifica os cálculos do caminho mais curto dos roteadores em outras áreas. Porém, se houver apenas um roteador de borda fora de uma área, nem mesmo o resumo precisa ser passado. As rotas para destinos fora da área sempre começam com a instrução "Vá até o roteador de borda". Esse tipo de área é chamado **área de stub**.

O último tipo de roteador é o **roteador de limite de AS**. Ele injeta na área as rotas para destinos externos em outros ASs. As rotas externas aparecem então como destinos que podem ser alcançados por meio do roteador de limite de AS com algum custo. Uma rota externa pode ser injetada em um ou mais roteadores de limite de AS. A relação entre ASs, áreas e os diversos tipos de roteadores aparece na Figura 5.56. Um roteador pode desempenhar vários papéis; por exemplo, um roteador de borda também é um roteador de backbone.

Durante a operação normal, cada roteador dentro de uma área tem o mesmo banco de dados de estado de enlace e executa o mesmo algoritmo de caminho mais curto. Sua tarefa principal é calcular o caminho mais curto de si mesmo para cada um dos roteadores e para a rede no AS inteiro. Um roteador de borda de área precisa dos bancos de dados para todas as áreas às quais está conectado, e precisa executar o algoritmo do caminho mais curto para cada área separadamente.

Para uma origem e um destino na mesma área, a melhor rota intra-área (que se encontra totalmente dentro dela) é escolhida. Para uma origem e um destino em áreas diferentes, a rota interárea deve ir da origem para o backbone, atravessá-lo até a área de destino e depois até o destino. Esse algoritmo força uma configuração de estrela no OSPF, com o backbone sendo o hub e as outras áreas sendo as pontas. Como a rota com o menor custo é escolhida, os roteadores em diferentes partes da rede podem usar diferentes roteadores de borda de área para entrar no backbone e na área de destino. Os pacotes são roteados da origem ao destino "como se encontram". Eles não são encapsulados ou tunelados (a menos que sigam para uma área cuja única conexão para o backbone é um túnel). Além disso, as rotas para destinos externos podem incluir um custo externo do roteador de limite do AS pelo caminho externo, se desejado, ou apenas o custo interno para o AS.

Quando um roteador é iniciado, ele envia mensagens HELLO em todas as conexões ponto a ponto e as transmite por multicast nas LANs para o grupo consistindo em todos os outros roteadores. A partir das respostas, cada roteador descobre quem são seus vizinhos. Os roteadores na mesma LAN são todos os vizinhos.

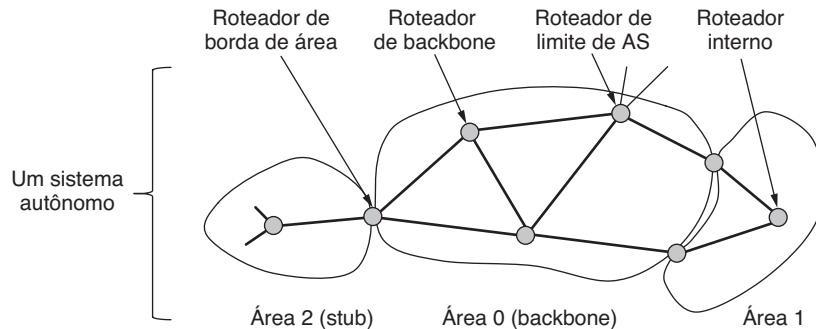



Figura 5.56 | A relação entre ASs, backbones e áreas no OSPF.

O OSPF funciona trocando informações entre roteadores adjacentes, o que não é o mesmo que entre roteadores vizinhos. Em particular, é ineficaz fazer com que cada roteador em uma LAN fale com outro roteador na LAN. Para evitar essa situação, um roteador é eleito como **roteador designado**. Ele é considerado **adjacente** a todos os outros em sua LAN, e troca informações com eles. Com efeito, ele está atuando como o único nó que representa a LAN. Os roteadores vizinhos que não são adjacentes não trocam informações uns com os outros. Um roteador designado de backup sempre é mantido atualizado, para facilitar a transição caso o roteador designado principal falhe e precise ser substituído imediatamente.

Durante a operação normal, cada roteador periodicamente envia mensagens LINK STATE UPDATE para cada um de seus roteadores adjacentes. Essas mensagens dão seu estado e oferecem os custos usados no banco de dados topológico. Essas mensagens de flooding são confirmadas, para torná-las confiáveis. Cada mensagem tem um número de sequência, de modo que um roteador pode ver se um LINK STATE UPDATE é mais antigo ou mais novo do que o que ele tem atualmente. Os roteadores também enviam essas mensagens quando um enlace é ativado ou desativado, ou quando seu custo muda.

Mensagens DATABASE DESCRIPTION dão os números de sequência de todas as entradas de estado de enlace atualmente mantidas pelo transmissor. Comparando seus próprios valores com os do transmissor, o receptor pode determinar quem tem os valores mais recentes. Essas mensagens são usadas quando um enlace é ativado.

Qualquer parceiro pode solicitar informações de estado de enlace do outro, usando mensagens LINK STATE REQUEST. O resultado desse algoritmo é que cada par de roteadores adjacentes verifica quem tem os dados mais recentes, e novas informações são espalhadas pela área dessa maneira. Todas essas mensagens são enviadas diretamente em pacotes IP. Os cinco tipos de mensagens são resumidos na Tabela 5.9.

Finalmente, podemos juntar todas as partes. Usando flooding, cada roteador informa a todos os outros roteadores em sua área sobre seus enlaces para outros roteadores e redes e o custo desses enlaces. Essa informação permite que cada roteador construa o grafo para sua(s) área(s) e calcule os caminhos mais curtos. A área de backbone também faz esse trabalho. Além disso, os roteadores de backbone aceitam informações dos roteadores de borda de área, a fim de calcular a melhor rota a partir de cada roteador de backbone para cada um dos outros roteadores. Essa informação é propagada de volta para os roteadores de borda de área, que a anunciam dentro de suas áreas. Usando essa informação, os roteadores internos podem selecionar a melhor rota para um destino fora de sua área, incluindo o melhor roteador de saída para o backbone.

Tipo de mensagem	Descrição
Hello	Usada para descobrir quem são os vizinhos
Link state update	Oferece os custos do transmissor aos seus vizinhos
Link state ack	Confirma a atualização do estado de enlace
Database description	Anuncia quais atualizações o transmissor tem
Link state request	Solicita informações do parceiro

Tabela 5.9 | Os cinco tipos de mensagens OSPF.

5.6.7 BGP — PROTOCOLO DE ROTEAMENTO DE GATEWAY EXTERIOR

Dentro de um único AS, OSPF e IS-IS são os protocolos normalmente utilizados. Entre os ASs, um protocolo diferente, chamado **BGP** (**Border Gateway Protocol**), é utilizado. Um protocolo diferente é necessário porque os objetivos de um protocolo intradomínio e de um interdomínio não são os mesmos. Tudo o que um protocolo intradomínio tem a fazer é mover pacotes da forma mais eficiente possível da origem ao destino. Ele não precisa se preocupar com a política.

Ao contrário, os protocolos de roteamento interdomínio precisam se preocupar muito com a política (Metz, 2001). Por exemplo, um AS corporativo poderia desejar a capacidade de enviar pacotes e recebê-los de qualquer site da Internet. Entretanto, pode não ser interessante transportar pacotes de trânsito começando em um AS estrangeiro e terminando em um AS estrangeiro diferente, mesmo que seu próprio AS esteja no caminho mais curto entre os dois ASs estrangeiros ("Isso é problema seu, e não nosso"). Por outro lado, pode ser interessante transportar o tráfego para seus vizinhos, ou mesmo para outros ASs específicos que pagaram por esse serviço. Companhias telefônicas, por exemplo, podem ter interesse em atuar como operadoras para seus clientes, mas não para outros. Os protocolos de gateway exterior em geral, e o BGP em particular, têm sido designados para permitir que muitos tipos de políticas de roteamento sejam impostos no tráfego entre ASs.

As políticas típicas envolvem considerações políticas, de segurança ou econômicas. Alguns possíveis exemplos de restrições de roteamento são:

1. Não transporte tráfego comercial na rede educacional.
2. Nunca envie tráfego do Pentágono em uma rota que passa pelo Iraque.
3. Use TeliaSonera em vez de Verizon, pois é mais barato.
4. Não use AT&T na Austrália, porque o desempenho é fraco.
5. O tráfego começando ou terminando na Apple não deve transitar pelo Google.

Como você pode imaginar por essa lista, as políticas de roteamento podem ser altamente individuais. Elas normalmente são patenteadas, pois contêm informações comerciais confidenciais. Porém, podemos descrever alguns padrões que capturam o raciocínio da empresa mencionada anteriormente e que normalmente são usados como ponto de partida.

Uma política de roteamento é implementada decidindo que tráfego pode fluir por quais dos enlaces entre os ASs. Uma política comum é que um ISP cliente paga a outro ISP provedor para entregar pacotes a qualquer outro destino na Internet e recebe pacotes enviados de qualquer outro destino. O ISP cliente compra **serviço de trânsito** do ISP provedor. Isso é exatamente como um cliente em casa comprando serviço de acesso à Internet de um ISP. Para fazer com que isso funcione, o provedor deve anunciar as rotas para todos os destinos na Internet ao cliente pelo enlace que os conecta. Desse modo, o cliente terá uma rota a ser usada para enviar pacotes a qualquer lugar. Reciprocamente, o cliente deverá anunciar rotas somente para os destinos em sua rede até o provedor. Isso permitirá que o provedor envie tráfego para o cliente somente para esses endereços; o cliente não deseja tratar do tráfego intencionado para outros destinos.

Podemos ver um exemplo de serviço de trânsito na Figura 5.57. Nela, existem quatro ASs que estão conectados. A conexão normalmente é feita com um enlace nos **pontos de troca da Internet**, ou **IXPs** (**Internet eXchange Points**), instalações em que muitos ISPs têm um enlace com a finalidade de se conectar a outros ISPs. *AS2*, *AS3* e *AS4* são clientes de *AS1*. Eles compram serviço de trânsito dele. Assim, quando a origem *A* envia para o destino *C*, os pacotes atravessam de *AS2* para *AS1* e, finalmente, para *AS4*. Os anúncios de roteamento transitam no sentido oposto ao dos pacotes. *AS4* anuncia *C* como um destino para seu provedor de trânsito, *AS1*, para permitir que as origens alcancem *C* por meio de *AS1*. Mais tarde, *AS1* anuncia uma rota para *C* a seus outros clientes, incluindo *AS2*, para permitir que os clientes saibam que eles podem enviar tráfego para *C* por meio de *AS1*.

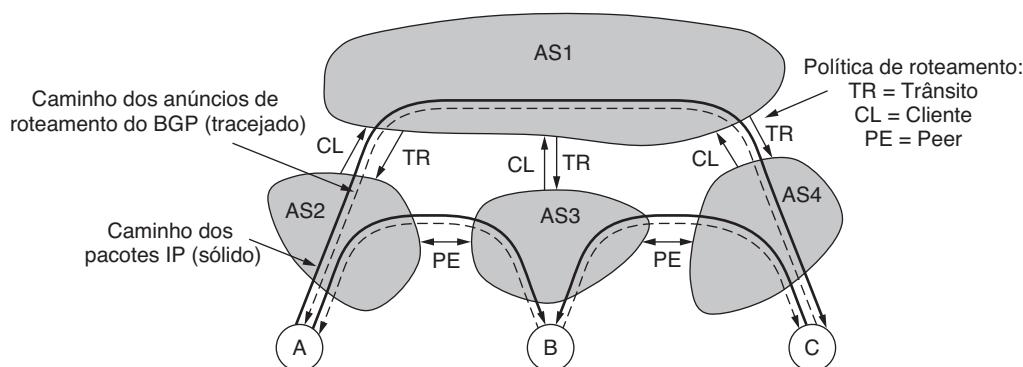



Figura 5.57 | Políticas de roteamento entre quatro sistemas autônomos.

Na Figura 5.57, todos os outros ASs compram serviço de trânsito de *AS1*. Isso lhes oferece conectividade, de modo que possam interagir com qualquer host na Internet. Porém, eles precisam pagar por esse privilégio. Suponha que *AS2* e *AS3* troquem muito tráfego. Visto que suas redes já estão conectadas, se eles quiserem, podem usar uma política diferente — eles podem enviar tráfego diretamente de um para outro gratuitamente. Isso reduzirá a quantidade de tráfego que *AS1* precisa entregar em seu favor, e espera-se que se reduzam suas contas. Essa política é chamada **peering**.

Para implementar o peering, dois ASs enviam anúncios de roteamento um para o outro, para os endereços que residem em suas redes. Isso torna possível que *AS2* envie para *AS3* pacotes de *A* destinados para *B* e vice-versa. Porém, observe que o peering não é transitivo. Na Figura 5.57, *AS3* e *AS4* também fazem par um com o outro. Esse peering permite que o tráfego de *C* destinado para *B* seja enviado diretamente para *AS4*. O que acontece se *C* envia um pacote para *A*? *AS3* só está anunciando uma rota para *B* para *AS4*. Ele não está anunciando uma rota para *A*. A consequência é que o tráfego não passará de *AS4* para *AS3* para *AS2*, embora exista um caminho físico. Essa restrição é exatamente o que *AS3* deseja. Ele faz par com *AS4* para trocar tráfego, mas não deseja transportar o tráfego de *AS4* para outras partes da Internet, pois não está sendo pago para fazer isso. Em vez disso, *AS4* recebe serviço de trânsito de *AS1*. Assim, é *AS1* quem transportará o pacote de *C* para *A*.

Agora que sabemos a respeito de trânsito e peering, também podemos ver que *A*, *B* e *C* têm arranjos de trânsito. Por exemplo, *A* precisa comprar acesso à Internet de *AS2*. *A* poderia ser um único computador doméstico ou uma rede de empresa com muitas LANs. Porém, ele não precisa rodar BGP, pois é uma **rede stub** conectada ao restante da Internet por apenas um enlace. Assim, o único lugar para ela enviar pacotes destinados para fora da rede é pelo enlace com *AS2*. Não há outro lugar para ir. Esse caminho pode ser arranjado simplesmente pela criação de uma rota-padrão. Por esse motivo, não mostramos *A*, *B* e *C* como ASs que participam no roteamento interdomínio.

Por outro lado, algumas redes de empresas são conectadas a vários ISPs. Essa técnica é usada para melhorar a confiabilidade, pois, se o caminho por um ISP falhar, a empresa pode usar o caminho por outro ISP. Essa técnica é chamada **multihoming**. Nesse caso, a rede da empresa provavelmente executará um protocolo de roteamento interdomínio (por exemplo, BGP) para dizer a outros ASs quais endereços devem ser alcançados por meio de quais enlaces de ISP.

São possíveis muitas variações sobre essas políticas de trânsito e peering, mas elas já ilustram como os relacionamentos e o controle da empresa sobre o local onde os anúncios de rota se encontram podem implementar diferentes tipos de políticas. Agora, vamos considerar com mais detalhes como os roteadores executando BGP anunciam rotas uns para os outros e selecionam caminhos sobre os quais os pacotes são encaminhados.

BGP é uma forma de protocolo por vetor de distância, mas é muito diferente dos protocolos por vetor de distância intradomínio, como RIP. Já vimos que essa política, em vez da distância mínima, é usada para escolher quais rotas utilizar. Outra grande diferença é que, em vez de manter apenas o custo da rota para cada destino, cada roteador BGP registra o caminho utilizado. Essa técnica é chamada **protocolo por vetor de caminho**. O caminho consiste no roteador do próximo hop (que pode estar no outro lado do ISP, e não adjacente) e na sequência de ASs, ou **caminho do AS**, que a rota seguiu (dado na ordem inversa). Finalmente, pares de roteadores BGP se comunicam uns com os outros estabelecendo conexões TCP. Operar dessa maneira oferece comunicação confiável e também oculta todos os detalhes da rede sendo atravessada.

Um exemplo de como as rotas BGP são anunciadas aparece na Figura 5.58. Existem três ASs, e o do meio está fornecendo trânsito para os ISPs da esquerda e da direita. Um anúncio de rota para o prefixo *C* começa em *AS3*. Quando propagado pelo enlace para *R2c* no alto da figura, ele tem o caminho de AS de simplesmente *AS3* e o roteador do próximo hop de *R3a*. Na parte de baixo, ele tem o mesmo caminho de AS, mas um próximo hop diferente, pois veio por um enlace diferente. Esse anúncio continua a se propagar e atravessa o limite para *AS1*. No roteador *R1a*, no alto da figura, o caminho do AS é *AS2*, *AS3* e o próximo hop é *R2a*.

Transportar o caminho completo com a rota a ser seguida torna mais fácil para o roteador receptor detectar e acabar com os loops de roteamento. Uma regra é que cada roteador que envia uma rota para fora do AS inicia seu próprio número de AS para a rota. (É por isso que a lista está em ordem invertida.) Quando um roteador recebe uma rota, ele verifica se seu próprio número de AS já está no caminho do AS. Se estiver, um loop foi detectado e o anúncio é descartado. Porém, e de certa forma ironicamente, no fim da década de 90 observou-se que, apesar dessa precaução, o BGP sofre de uma versão do problema da contagem ao infinito (Labovitz et al., 2001). Não existem loops de longa vida, mas as rotas às vezes podem demorar a convergir e ter loops transitórios.

Oferecer uma lista de ASs é um modo muito primitivo de especificar um caminho. Um AS poderia ser uma pequena empresa, ou uma rede de backbone internacional. Não dá para saber isso pela rota. O BGP nem sequer tenta, pois diferentes ASs podem usar diferentes protocolos intradomínio cujos custos podem ser comparados. Mesmo que não pudessem ser comparados, um AS pode não querer revelar suas métricas internas. Essa é uma das maneiras pelas quais os protocolos de roteamento interdomínio diferem dos intradomínio.

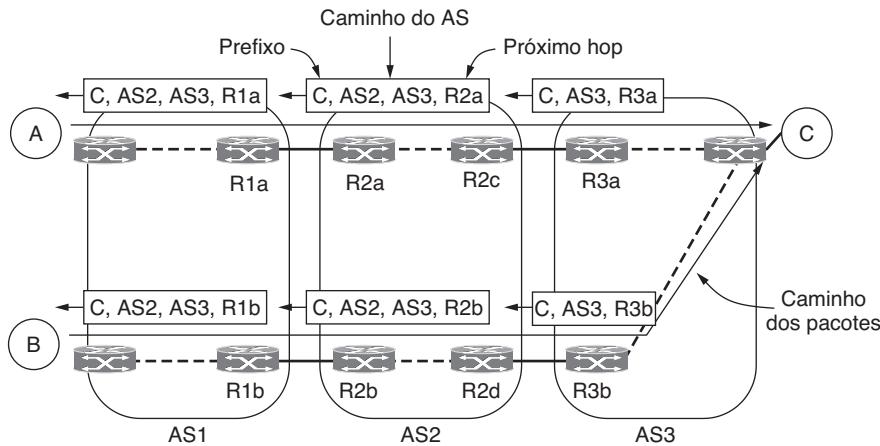



Figura 5.58 | Propagação dos anúncios de rota BGP.

Até aqui, vimos como um anúncio de rota é enviado pelo enlace entre dois ISPs. Ainda precisamos de alguma maneira para propagar as rotas BGP de um lado do ISP para o outro, de modo que possam ser enviadas para o próximo ISP. Essa tarefa poderia ser tratada pelo protocolo intradomínio, mas, como o BGP é muito bom na expansão para redes grandes, uma variante do BGP normalmente é utilizada. Ela é chamada **BGP interno**, ou **iBGP** (**internal BGP**), para distingui-la do uso regular do BGP, como **BGP externo**, ou **eBGP** (**external BGP**).

A regra para propagar rotas dentro de um ISP é que cada roteador no limite do ISP descobre todas as rotas vistas por todos os outros roteadores de limite, por consistência. Se um roteador de limite no ISP descobrir um prefixo para o IP 128.208.0.0/16, todos os outros roteadores descobrirão esse prefixo. Este poderá, então, ser alcançado a partir de todas as partes do ISP, não importando como os pacotes entram no ISP a partir de outros ASs.

Não mostramos essa propagação na Figura 5.58 para evitar confusão, mas, por exemplo, o roteador *R2b* saberá que pode alcançar *C* por meio do roteador *R2c* no alto ou do roteador *R2d* na parte de baixo. O próximo hop é atualizado à medida que a rota cruza dentro do ISP, de modo que os roteadores no lado distante do ISP sabem qual roteador usar para sair do ISP no outro lado. Isso pode ser visto nas rotas mais à esquerda, em que o próximo hop aponta para um roteador no mesmo ISP, e não para um roteador no próximo ISP.

Agora, podemos descrever a principal parte que falta, que é o modo como os roteadores BGP escolhem a rota a utilizar para cada destino. Cada roteador BGP pode descobrir uma rota para determinado destino a partir do roteador ao qual está conectado no próximo ISP e a partir de todos os outros roteadores no limite (que escutaram diferentes rotas dos roteadores aos quais estão conectados em outros ISPs). Cada roteador precisa decidir qual rota nesse conjunto é a melhor para usar. Por fim, a resposta é que fica a critério do ISP escrever alguma política para escolher a rota preferida. Porém, essa explicação é muito genérica e não satisfaz de forma alguma, de modo que podemos pelo menos descrever algumas estratégias comuns.

A primeira estratégia é que as rotas por redes peering são escolhidas em preferência às rotas por meio de provedores de acesso. As primeiras são gratuitas; as últimas têm um custo monetário. Uma estratégia semelhante é que as rotas do cliente recebem preferência mais alta. Só é bom negócio enviar o tráfego diretamente para os clientes que estão pagando.

Um tipo diferente de estratégia é a regra-padrão de que os caminhos de AS mais curtos são melhores. Isso é discutível, pois um AS poderia ser uma rede de qualquer tamanho, de modo que o caminho por três ASs pequenos poderia realmente ser mais curto do que um caminho por um AS grande. Porém, na média, mais curto costuma ser melhor, e essa regra é um critério de desempate comum.

A estratégia final é preferir a rota que tem o menor custo dentro do ISP. Essa é a estratégia implementada na Figura 5.58. Os pacotes enviados de *A* para *C* saem de *AS1* no roteador de cima, *R1a*. Os pacotes enviados de *B* saem pelo roteador de baixo, *R1b*. O motivo é que tanto *A* quanto *B* estão tomando o caminho com menor custo ou a rota mais rápida para fora de *AS1*. Por estarem eles localizados em partes diferentes do ISP, a saída mais rápida para cada um é diferente. A mesma coisa acontece quando os pacotes passam por *AS2*. Na última perna, *AS3* precisa transportar o pacote de *B* passando por sua própria rede.

Essa estratégia é conhecida como **saída antecipada** ou **roteamento 'batata quente'**. Ela tem o efeito colateral curioso de tender a tornar as rotas assimétricas. Por exemplo, considere o caminho seguido quando *C* envia um pacote de volta para *B*. O pacote sairá de *AS3* rapidamente, no roteador de cima, para evitar desperdiçar seus recursos. De modo semelhante, ele permanecerá no topo quando *AS2* lhe passar para *AS1* o mais rapidamente possível. De-

pois, o pacote terá uma viagem mais longa em *AS1*. Essa é a imagem-espelho do caminho seguido de *B* até *C*.

Essa discussão deverá deixar claro que cada roteador BGP escolhe sua própria melhor rota a partir das possibilidades conhecidas. Não acontece, como se poderia esperar de forma ingênua, que o BGP escolhesse um caminho a seguir no nível do AS e o OSPF escolhesse caminhos dentro de cada um dos ASs. O BGP e o protocolo de gateway interior estão muito mais profundamente integrados. Isso significa que, por exemplo, o BGP pode encontrar o melhor ponto de saída de um ISP para o próximo nesse ponto, e variará pelo ISP, como no caso da política 'batata quente'. Isso também significa que os roteadores BGP em diferentes partes de um AS podem escolher diferentes caminhos de AS para alcançar o mesmo destino. O ISP deve ter o cuidado de configurar todos os roteadores BGP para fazer escolhas compatíveis, dada toda essa liberdade, mas isso pode ser feito na prática.

O interessante é que só arranhamos a superfície do BGP. Para obter mais informações, consulte a especificação do BGP versão 4 na RFC 4271 e nas RFCs relacionadas. Porém, observe que grande parte de sua complexidade está nas políticas, as quais não são descritas na especificação do protocolo BGP.

5.6.8 Multicast na Internet

A comunicação IP normal é feita entre um transmissor e um receptor. Porém, para algumas aplicações, é útil que um processo seja capaz de enviar dados para um grande número de receptores simultaneamente. Alguns exemplos são o streaming de um evento esportivo ao vivo para muitos espectadores, oferecendo atualizações de programa a um pool de servidores replicados, e tratando de chamadas telefônicas de conferência digital (ou seja, em múltiplas partes).

O IP oferece suporte para a comunicação um para muitos, ou multicasting, usando endereços IP de classe D. Cada endereço de classe D identifica um grupo de hosts. Vinte e oito bits estão disponíveis para identificar os grupos, de modo que mais de 250 milhões de grupos podem existir ao mesmo tempo. Quando um processo envia um pacote para um endereço de classe D, é feita uma tentativa pelo melhor esforço para entregá-lo a todos os membros do grupo endereçado, mas nenhuma garantia é dada. Alguns membros podem não receber o pacote.

A faixa de endereços IP 224.0.0.0/24 é reservada para multicast na rede local. Nesse caso, nenhum protocolo de roteamento é necessário. Os pacotes são transmitidos por multicast simplesmente transmitindo-os na LAN com endereços de multicast. Todos os hosts na LAN recebem os broadcasts, e os hosts que são membros do grupo processam o pacote. Os roteadores não encaminham o pacote pela LAN. Alguns exemplos dos endereços de multicast locais são:

224.0.0.1	Todos os sistemas em uma LAN
224.0.0.2	Todos os roteadores em uma LAN
224.0.0.5	Todos os roteadores OSPF em uma LAN
224.0.0.251	Todos os servidores de DNS em uma LAN

Outros endereços de multicast podem ter membros em redes diferentes. Nesse caso, um protocolo de roteamento é necessário. Mas, primeiro, os roteadores de multicast precisam saber quais hosts são membros de um grupo. Um processo pede ao seu host para se juntar a um grupo específico. Ele também pode pedir a seu host para sair do grupo. Cada host registra a quais grupos seus processos pertencem atualmente. Quando o último processo em um host sai de um grupo, ele não é mais um membro desse grupo. Cerca de uma vez por minuto, cada roteador multicast envia um pacote de consulta a todos os hosts em sua LAN (usando o endereço de multicast local 224.0.0.1, naturalmente), pedindo a eles que informem de volta os grupos aos quais pertencem atualmente. Os roteadores de multicast podem ou não estar localizados com os roteadores-padrão. Cada host envia respostas de volta para todos os endereços de classe D em que está interessado. Esses pacotes de consulta e resposta utilizam um protocolo chamado **protocolo de gerenciamento de grupo da Internet**, ou **IGMP** (**Internet Group Management Protocol**). Este é descrito na RFC 3376.

Qualquer um dos diversos protocolos de roteamento multicast pode ser usado para criar spanning trees de multicast que oferecem caminhos dos transmissores para todos os membros do grupo. Os algoritmos usados são aqueles que descrevemos na Seção 5.2.8. Dentro do AS, o protocolo principal usado é o **multicast independente de protocolo**, ou **PIM** (**Protocol Independent Multicast**). O PIM pode ser de vários tipos. No Dense Mode PIM, cria-se uma walk tree podada de caminho reverso. Isso é adequado para situações em que os membros estão em toda a parte da rede, como ao distribuir arquivos para muitos servidores dentro da rede de um centro de dados. No Sparse Mode PIM, as spanning trees criadas são semelhantes às árvores baseadas em núcleo. Isso é adequado para situações como um provedor de conteúdo realizando o multicasting de TV para os assinantes em sua rede IP. Uma variante desse projeto, chamada Source-Specific Multicast PIM, é otimizada para o caso em que existe apenas um transmissor para o grupo. Finalmente, as extensões do multicast ao BGP ou túneis precisam ser usadas para criar rotas multicasting quando os membros do grupo estão em mais de um AS.

5.6.9 IP móvel

Muitos usuários da Internet têm computadores móveis e querem permanecer conectados quando estão longe de casa e até mesmo na estrada. Infelizmente, o sistema de endereçamento IP torna o trabalho longe de casa muito mais fácil de falar do que de fazer, como descreveremos

em breve. Quando as pessoas começaram a exigir a capacidade de alguma maneira, a IETF estabeleceu um grupo de trabalho para encontrar uma solução. O grupo de trabalho rapidamente formulou uma série de objetivos considerados desejáveis em qualquer solução. Os principais foram:

1. Cada host móvel precisa ser capaz de usar seu endereço IP doméstico de qualquer lugar.
2. Mudanças de software nos hosts fixos não foram permitidas.
3. Mudanças no software do roteador e nas tabelas não foram permitidas.
4. A maioria dos pacotes para hosts móveis não deveria fazer desvios no caminho.
5. Nenhum overhead deve ser contraído quando um host móvel está em casa.

A solução escolhida foi a descrita na Seção 5.2.10. Resumindo, cada local que deseja permitir que seus usuários viajem precisa criar um auxiliador no local, chamado **agente de origem**. Quando um host móvel aparece em um local externo, ele obtém um novo endereço IP (chamado endereço care of) no local externo. Então, o móvel diz ao agente de origem onde ele se encontra naquele momento, dando-lhe o endereço care of. Quando um pacote para o móvel chega ao local de origem e o móvel está em outro lugar, o agente de origem apanha o pacote e o envia por um túnel ao móvel, no endereço care of atual. O móvel pode enviar pacotes de resposta diretamente para qualquer um com quem esteja se comunicando, mas ainda usando seu endereço de origem. Essa solução reúne todos os requisitos apresentados anteriormente, exceto que os pacotes para hosts móveis fazem desvios.

Agora que cobrimos a camada de rede da Internet, podemos entrar na solução com mais detalhes. A necessidade de suporte para mobilidade em primeiro lugar vem do próprio esquema de endereçamento IP. Cada endereço IP contém um número de rede e um número de host. Por exemplo, considere a máquina com endereço IP 160.80.40.20/16. O 160.80 indica o número da rede; 40.20 é o número do host. Os roteadores do mundo inteiro possuem tabelas de roteamento informando qual enlace usar para chegar à rede 160.80. Sempre que um pacote chega com um endereço IP de destino na forma 160.80.xx.yy, ele sai dessa interface. Se, de repente, a máquina com esse endereço é enviada para algum local distante, os pacotes para ela continuarão a ser roteados para sua LAN (ou roteador) de origem.

Nesse estágio, há duas opções — ambas não atraentes. A primeira é que poderíamos criar uma rota a um prefixo mais específico. Ou seja, se o local distante anunciar uma rota para 160.80.40.20/32, os pacotes enviados para o destino começarão a chegar ao local correto novamente. Essa opção depende do algoritmo com maior prefixo combinado que é usado nos roteadores. Porém, acrescentamos uma rota para um prefixo IP com um único endereço IP nela. Todos os ISPs no mundo descobrirão sobre esse prefixo. Se todos mudarem as rotas IP globais dessa maneira quando mudam seu computador de local, cada roteador teria milhões de entradas na tabela, a um custo astronômico para a Internet. Essa opção não é funcional.

A segunda opção é mudar o endereço IP do dispositivo móvel. É verdade que os pacotes enviados para o endereço IP de origem não serão mais entregues até que todas as pessoas, programas e bancos de dados relevantes sejam informados da mudança. Mas o dispositivo móvel ainda poderá usar a Internet como o novo local para navegar pela Web e executar outras aplicações. Essa opção trata da mobilidade em uma camada mais alta. Isso é o que normalmente acontece quando um usuário leva um notebook a uma lanchonete e usa a Internet por meio da rede wireless local. A desvantagem é que isso impede algumas aplicações e não mantém a conectividade enquanto o dispositivo móvel se movimenta.

A propósito, a mobilidade também pode ser tratada em uma camada mais baixa, a camada de enlace. É isso que acontece quando se usa um notebook em uma única rede wireless 802.11. O endereço IP do dispositivo móvel não muda e o caminho da rede permanece o mesmo. É o enlace sem fios que está oferecendo mobilidade. Porém, o grau de mobilidade é limitado. Se o notebook se mover para muito longe, ele terá de se conectar à Internet por meio de outra rede, com um endereço IP diferente.

A solução do IP móvel para IPv4 é dada na RFC 3344. Ela funciona com o roteamento de Internet existente e permite que os hosts permaneçam conectados com seus próprios endereços IP enquanto se movimentam. Para que isso funcione, o dispositivo móvel precisa ser capaz de descobrir quando ele foi movido. Isso é feito com mensagens ICMP de anúncio e solicitação do roteador. Os dispositivos móveis escutam anúncios periódicos do roteador ou enviam uma solicitação para descobrir o mais próximo. Se esse roteador não for o endereço normal do roteador quando o dispositivo móvel estiver em casa, ele precisa estar em uma rede externa. Se esse roteador tiver mudado desde a última vez, o dispositivo móvel terá passado para outra rede externa. Esse mesmo mecanismo permite que os hosts móveis encontrem seus agentes de origem.

Para obter um endereço IP care of na rede externa, um dispositivo móvel pode simplesmente usar o DHCP. Como alternativa, se os endereços IPv4 forem escassos, o dispositivo móvel pode enviar e receber pacotes por meio de um agente externo que já tem um endereço IP na rede. O host móvel encontra um agente externo usando o mesmo mecanismo ICMP adotado para encontrar o agente de origem. Depois que o dispositivo móvel obtém um endereço IP ou encontra um agente externo, ele é capaz de usar a rede para enviar uma mensagem a seu agente de origem, informando-lhe sobre seu local atual.

O agente de origem precisa de uma maneira de interceptar pacotes enviados ao dispositivo móvel somente quando este não estiver na origem. O ARP oferece um mecanismo conveniente. Para enviar um pacote por uma Ethernet a um host IP, o roteador precisa saber o endereço Ethernet do host. O mecanismo normal é que o roteador envie uma consulta ARP para perguntar, por exemplo, qual é o endereço Ethernet de 160.80.40.20. Quando o dispositivo móvel está na origem, ele responde às consultas ARP por seu endereço IP com seu próprio endereço Ethernet. Quando o dispositivo móvel está longe, o agente de origem responde a essa consulta oferecendo seu endereço Ethernet. O roteador, então, envia pacotes para 160.80.40.20 ao agente de origem. Lembre-se de que isso é chamado de proxy ARP.

Para atualizar rapidamente os mapeamentos ARP de um lado para outro quando o dispositivo móvel sai da origem ou chega de volta, pode ser usada outra técnica ARP, chamada **ARP gratuito**. Basicamente, o agente móvel ou de origem envia ele mesmo uma consulta ARP para o endereço IP do dispositivo móvel, que fornece a resposta certa, de modo que o roteador observa e atualiza seu mapeamento.

O tunelamento para enviar um pacote entre o agente de origem e o host móvel no endereço care of é feito encapsulando o pacote com outro cabeçalho IP destinado para o endereço care of. Quando o pacote encapsulado chega ao endereço care of, o cabeçalho IP externo é removido para revelar o pacote.

Como em muitos protocolos da Internet, o problema está nos detalhes, e geralmente os detalhes de compatibilidade com outros protocolos que estão implementados. Existem duas complicações. Primeiro, os NATs examinam, além do cabeçalho IP, os cabeçalhos TCP ou UDP. A forma original de tunelamento para o IP móvel não usava esses cabeçalhos, de modo que não funcionava com NATs. A solução foi mudar o encapsulamento para incluir um cabeçalho UDP.

A segunda complicação é que alguns ISPs verificam os endereços IP de origem dos pacotes para ver se eles combinam com o local onde o protocolo de roteamento acredita que a origem deva estar localizada. Essa técnica é chamada **filtragem de acesso**, e é uma medida de segurança com a intenção de descartar o tráfego, com endereços aparentemente incorretos, que possa ser malicioso. Porém, os pacotes enviados do dispositivo móvel para outros hosts da Internet quando ele está em uma rede externa terão um endereço IP que está fora de lugar, de modo que serão descartados. Para contornar esse problema, o dispositivo móvel pode usar o endereço care of como uma origem para tunelar os pacotes de volta a seu agente de origem. Daqui, eles são enviados para a Internet a partir do que parece ser o local correto. O custo é que a rota é mais longa.

Outra questão que ainda não discutimos é a segurança. Quando um agente de origem recebe uma mensagem pedindo que ele, por favor, encaminhe todos os pacotes de Roberta para algum endereço IP, teria sido melhor não concordar, a menos que esteja convicto de que Roberta realmente é a origem dessa solicitação, e não alguém tentando se passar por ela. Os protocolos de autenticação criptográfica são usados para essa finalidade. Estudaremos esses protocolos no Capítulo 8.

Os protocolos de mobilidade para o IPv6 se basearam no alicerce do IPv4. O esquema apresentado sofre com o problema do roteamento triangular, em que os pacotes enviados ao dispositivo móvel seguem uma rota por um agente de origem distante. No IPv6, a otimização de rota é usada para seguir um caminho direto entre o dispositivo móvel e outros endereços IP após os pacotes iniciais terem seguido a rota mais longa. O IPv6 móvel é definido na RFC 3775.

Ainda existe outro tipo de mobilidade que também está sendo definido para a Internet. Alguns aviões possuem redes wireless embutidas, as quais os passageiros podem usar para conectar seus notebooks à Internet. O avião possui um roteador que se conecta ao restante da Internet por meio de um enlace wireless. (Você esperava um enlace com fios?) Assim, agora temos um roteador em voo, o que significa que a rede inteira é móvel. Os projetos de mobilidade da rede aceitam essa situação sem que os notebooks percebam que o avião é móvel. Com relação a eles, essa é apenas outra rede. Naturalmente, alguns dos notebooks podem estar usando o IP móvel para manter seus endereços de origem enquanto estão no avião, de modo que temos dois níveis de mobilidade. A mobilidade da rede é definida para o IPv6 na RFC 3963.

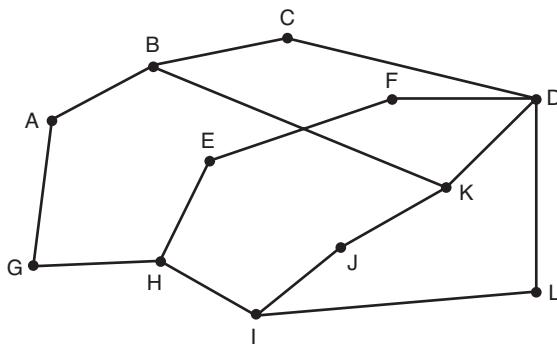
5.7 Resumo

A camada de rede fornece serviços à camada de transporte. Ela pode se basear em circuitos virtuais ou em datagramas. Em ambos os casos, sua principal tarefa é rotear pacotes da origem até o destino. Nas redes de datagramas, uma decisão de roteamento é tomada para cada pacote. Nas redes de circuitos virtuais, ela é tomada quando o circuito virtual é estabelecido.

Muitos algoritmos de roteamento são usados nas redes de computadores. O flooding é um algoritmo simples para enviar um pacote por todos os caminhos. A maioria dos algoritmos encontra o caminho mais curto e se adapta às mudanças na topologia da rede. Os algoritmos principais são o roteamento por vetor de distância e o roteamento de estado de enlace. A maioria das redes reais utiliza um desses algoritmos. Outros assuntos importantes relacionados ao roteamento são o uso de hierarquia em grandes redes, o roteamento de hosts móveis e o roteamento por broadcast, multicast e anycast.

As redes podem facilmente se tornar congestionadas, aumentando o atraso e a perda de pacotes. Os projetistas de rede tentam evitar o congestionamento projetando-a para que tenha capacidade suficiente, escolhendo rotas não congestionadas, recusando-se a aceitar mais tráfego, sinalizando as origens para reduzir a velocidade e escoando a carga.

A próxima etapa, além de lidar com o congestionamento, é tentar de fato alcançar uma qualidade de serviço prometida. Algumas aplicações se importam mais com o throughput, enquanto outras se importam mais com o atraso e a flutuação. Os métodos que podem ser usados para fornecer diferentes qualidades de serviço incluem uma combinação de adequação de tráfego, reserva de recursos nos roteadores e controle de acesso. Entre as abordagens adotadas para obter boa qualidade de serviço estão os serviços integrados da IETF (incluindo RSVP) e os serviços diferenciados.

As redes apresentam diferenças em vários aspectos; portanto, podem ocorrer problemas quando várias redes estão interconectadas. Quando diversas redes possuem diferentes tamanhos máximos de pacote, a fragmentação pode ser necessária. Diferentes redes podem usar diferentes protocolos de roteamento internamente, mas precisam executar um protocolo comum externamente. Às vezes, os problemas podem ser superados efetuando-se o tunelamento quando um pacote passa por uma rede hostil, mas, se as redes de origem e de destino forem diferentes, essa estratégia não funcionará.

A Internet tem uma rica variedade de protocolos relacionados à camada de rede. Entre eles, encontram-se o protocolo de transporte de dados a qualquer custo, o IP e os protocolos de controle como ICMP, ARP e DHCP associados. Um protocolo orientado à conexão, chamado MPLS, transporta pacotes IP por algumas redes. Um dos principais protocolos de roteamento usados dentro das redes é o OSPF, e o protocolo de roteamento usado entre as redes é o BGP. A Internet está esgotando rapidamente os endereços IP, de modo que foi desenvolvida uma nova versão do IP, o IPv6, para resolver esse problema.

Problemas

1. Cite dois exemplos de aplicações para as quais é apropriado um serviço orientado a conexões. A seguir, cite dois exemplos para os quais é mais apropriado um serviço não orientado a conexões.
2. As redes de datagramas roteiam cada pacote como uma unidade separada, independente das demais. As redes de circuitos virtuais não precisam tomar essa providência, pois os pacotes de dados seguem uma rota predeterminada. Você diria que essa observação significa que as redes de circuitos virtuais não precisam da capacidade de rotear pacotes isolados de uma origem arbitrária até um destino arbitrário? Explique sua resposta.
3. Cite três exemplos de parâmetros de protocolos que devem ser negociados quando é configurada uma conexão.
4. Supondo que todos os roteadores e hosts estão funcionando adequadamente e todo o software está isento de todos os erros, há alguma chance, por menor que seja, de um pacote ser entregue no destino errado?
5. Cite uma heurística simples para localizar dois caminhos através de uma rede de determinada origem para determinado destino que possa sobreviver à perda de qualquer linha de comunicação (desde que existam dois desses caminhos). Os roteadores são considerados suficientemente confiáveis; portanto, não é preciso se preocupar com a possibilidade de panes em roteadores.
6. Considere a sub-rede da Figura 5.10(a). O roteamento por vetor de distância é usado e os seguintes vetores acabaram de entrar no roteador *C*: de *B*: (5, 0, 8, 12, 6, 2); de *D*: (16, 12, 6, 0, 9, 10); e de *E*: (7, 6, 3, 9, 0, 4). Os custos dos enlaces de *C* para *B*, *D* e *E* são 6, 3 e 5, respectivamente. Qual é a nova tabela de roteamento de *C*? Forneça a interface de saída a ser usada e o custo esperado.
7. Se os custos forem registrados como números de 8 bits em uma rede com 50 roteadores e os vetores de distância forem trocados duas vezes por segundo, qual será a largura de banda por linha (full-duplex) ocupada pelo algoritmo de roteamento distribuído? Parta do princípio de que cada roteador tem três linhas para os outros roteadores.
8. Na Figura 5.11, o OR booleano dos dois conjuntos de bits ACF é 111 em cada interface. Trata-se de uma coincidência ou ele é mantido em todas as sub-redes, independentemente das circunstâncias?
9. No roteamento hierárquico com 4.800 roteadores, que tamanhos de região e de agrupamento devem ser escolhidos para minimizar o tamanho da tabela de roteamento no caso de uma hierarquia de três camadas? Um bom ponto de partida é a hipótese de que uma solução com *k* agrupamentos de *k* regiões com *k* roteadores está próxima de ser ótima; isso significa que *k* é aproximadamente a raiz cúbica de 4.800 (cerca de 16). Utilize um processo de tentativa e erro para verificar as combinações em que todos os três parâmetros estão na vizinhança genérica de 16.
10. No texto foi declarado que, quando um host móvel não está em sua localização de origem, os pacotes enviados para sua LAN são interceptados pelo agente local dessa LAN. Em uma rede IP de uma LAN 802.3, de que maneira o agente local executa essa interceptação?
11. Observando a rede da Figura 5.5, quantos pacotes são gerados por broadcast de *B*, usando-se:
    (a) encaminhamento pelo caminho inverso?
    (b) a árvore de escoamento?

12. Considere a rede da Figura 5.13(a). Imagine que uma nova interface seja acrescentada entre *F* e *G*, mas que a árvore de escoamento da Figura 5.13(b) permaneça inalterada. Que mudanças ocorrerão na Figura 5.13(c)?
13. Calcule a spanning tree de multicast para o roteador *C* na rede a seguir para um grupo com membros nos roteadores *A*, *B*, *C*, *D*, *E*, *F*, *I* e *K*.

14. Suponha que o nó *B* na Figura 5.18 tenha acabado de ser reiniciado e não tenha nenhuma informação de roteamento em suas tabelas. De repente, ele precisa de uma rota para *H*. O nó transmite broadcasts com *TTL* definido como 1, 2, 3 e assim por diante. Quantas rodadas ele levará para encontrar uma rota?
15. Como um possível mecanismo de controle de congestionamento em uma rede que utiliza circuitos virtuais internamente, um roteador poderia privar-se de confirmar um pacote recebido até (1) saber que sua última transmissão ao longo do circuito virtual foi recebida com sucesso e (2) ter um buffer livre. Para simplificar, suponha que os roteadores usem um protocolo stop-and-wait e que cada circuito virtual tenha um buffer dedicado a ele em cada sentido do tráfego. Se ele precisar de *T* segundos para transmitir um pacote (dados ou confirmação) e houver *n* roteadores no caminho, qual será a taxa em que os pacotes serão entregues ao host de destino? Suponha que os erros de transmissão sejam raros e que a conexão entre host e roteador seja infinitamente rápida.
16. Uma rede de datagramas permite que os roteadores eliminem pacotes sempre que precisarem. A probabilidade de um roteador descartar um pacote é *p*. Considere o caso de um host de origem conectado ao roteador de origem, que está conectado ao roteador de destino que, por sua vez, está conectado ao host de destino. Se um dos roteadores descartar um pacote, o host de origem por fim sofrerá um timeout e fará novas tentativas. Se as interfaces host-roteador e roteador-roteador fossem contadas como hops, qual seria o número médio de:
    (a) hops que um pacote executa por transmissão?
    (b) transmissões que um pacote cria?
    (c) hops necessários por pacote recebido?
17. Descreva duas diferenças importantes entre o método ECN e o método RED de prevenção de congestionamento.
18. Um esquema token bucket é usado para a modelagem de tráfego. Um novo token é colocado no balde a cada 5 μs. Cada token é válido como um pacote curto, que contém 48 bytes de dados. Qual é a taxa máxima de dados sustentável?
19. Um computador em uma rede de 6 Mbps é controlado por um token bucket, o qual é preenchido a uma taxa de 1 Mbps. Inicialmente, sua capacidade é de 8 megabits. Durante quanto tempo o computador pode transmitir a 6 Mbps?
20. A rede da Figura 5.30 utiliza o RSVP com árvores de multicast nos hosts 1 e 2, como mostra a figura. Suponha que o host 3 solicite um canal de largura de banda de 2 MB/s para um fluxo proveniente do host 1 e outro canal de largura de banda de 1 MB/s para um fluxo do host 2. Ao mesmo tempo, o host 4 solicita um canal de largura de banda igual a 2 MB/s para um fluxo do host 1, e o host 5 solicita um canal de largura de banda de 1 MB/s para um fluxo vindo do host 2. Qual será a largura de banda total reservada para essas solicitações nos roteadores *A*, *B*, *C*, *E*, *H*, *J*, *K* e *L*?
21. Um roteador pode processar 2 milhões de pacotes/s. A carga oferecida a ele é de 1,5 milhão de pacotes/s em média. Se uma rota da origem até o destino contiver dez roteadores, quanto tempo será gasto no enfileiramento e no serviço pelo roteador?
22. Considere o usuário de serviços diferenciados com encaminhamento expresso. Existe alguma garantia de que os pacotes expressos experimentarão um atraso mais curto que os pacotes regulares? Por quê?
23. Suponha que o host *A* esteja conectado a um roteador *R*1, que *R*1 esteja conectado a outro roteador *R*2, e que *R2* esteja conectado ao host *B*. Suponha que uma mensagem TCP contendo 900 bytes de dados e 20 bytes de cabeçalho TCP seja repassada ao código IP do host *A* para ser entregue a *B*. Mostre os campos *Tamanho total*, *Identificação*, *DF*, *MF* e *Deslocamento de fragmento* do cabeçalho IP em cada pacote transmitido pelos três enlaces. Suponha que o enlace *A*-*R*1 possa admitir um tamanho máximo de quadro de 1.024 bytes, incluindo um cabeçalho de quadro de 14 bytes, que o enlace *R*1-*R*2 possa admitir um tamanho máximo de quadro de 512 bytes, incluindo um cabeçalho de quadro de 8 bytes, e que o enlace *R*2-*B* possa admitir um tamanho máximo de quadro de 512 bytes, incluindo um cabeçalho de quadro de 12 bytes.
24. Um roteador está transmitindo pacotes IP cujo comprimento total (dados mais cabeçalho) é de 1.024 bytes. Supondo que os pacotes tenham a duração de 10 segundos, qual será a velocidade máxima de linha em que o roteador poderá operar sem o perigo de circular pelo espaço de números de identificação de datagramas do IP?
25. Um datagrama IP usando a opção *Strict source routing* tem de ser fragmentado. Para você, a opção é copiada para cada fragmento ou basta colocá-la no primeiro fragmento? Explique sua resposta.
26. Suponha que, em vez de ser utilizados 16 bits na parte de rede de um endereço de classe B, tenham sido usados 20 bits. Nesse caso, existiram quantas redes da classe B?

27. Converta o endereço IP cuja representação hexadecimal é C22F1582 em uma notação decimal com pontos.

28. A máscara de sub-rede de uma rede na Internet é 255.255.240.0. Qual e o número máximo de hosts que ela pode manipular?

29 Ao passo que os endereços IP são usados para redes específicas, os endereços Ethernet não o são. Você consegue imaginar um bom motivo para isso?

30. Um grande número de endereços IP consecutivos está disponível a partir de 198.16.0.0. Suponha que quatro organizações, *A*, *B*, *C* e *D*, solicitem 4.000, 2.000, 4.000 e 8.000 endereços, respectivamente, e nessa ordem. Para cada uma delas, forneça o primeiro endereço IP atribuído, o último endereço IP atribuído e a máscara na notação *w.x.y.z/s*.

31. Um roteador acabou de receber estes novos endereços IP: 57.6.96.0/21, 57.6.104.0/21, 57.6.112.0/21 e 57.6.120.0/21. Se todos usarem a mesma interface de saída, eles poderão ser agregados? Em caso afirmativo, por quê? Em caso negativo, por quê?

32. O conjunto de endereços IP de 29.18.0.0 até 19.18.128.255 foi agregado a 29.18.0.0/17. Porém, existe um intervalo de 1.024 endereços não atribuídos, desde 29.18.60.0 até 29.18.63.255, que agora são repentinamente atribuídos a um host que utiliza uma interface de saída diferente. É necessário dividir o endereço agregado em seus blocos constituintes, acrescentar o novo bloco à tabela e, depois, verificar se é possível uma reagregação? Em caso contrário, o que poderia ser feito?

33. Um roteador tem as seguintes entradas (CIDR) em sua tabela de roteamento:

Endereço/máscara	Próximo hop
135.46.56.0/22	Interface 0
135.46.60.0/22	Interface 1
192.53.40.0/23	Roteador 1
padrão	Roteador 2

Para cada um dos endereços IP a seguir, o que o roteador fará se chegar um pacote com esse endereço?

(a) 135.46.63.10
(b) 135.46.57.14
(c) 135.46.52.2
(d) 192.53.40.7
(e) 192.53.56.7

34. Muitas empresas adotam a política de manter dois (ou mais) roteadores para conectar a empresa à Internet, a fim de proporcionar alguma redundância no caso de um deles ficar inativo. Essa política ainda será possível com a NAT? Explique sua resposta.

35. Você acabou de explicar o que é um protocolo ARP a um amigo. No final, ele diz: "Entendi. Como o ARP fornece um serviço à camada de rede, isso significa que ele faz parte da camada de enlace de dados". O que você diz a ele?

36. Descreva uma forma de remontar os fragmentos IP no destino.

37. A maioria dos algoritmos de remontagem do datagrama IP tem um timer para evitar que um fragmento perdido seja anexado definitivamente aos buffers de remontagem. Suponha que um datagrama tenha sido dividido em quatro fragmentos. Os três primeiros fragmentos chegam, mas o último deles está atrasado. A certa altura, o timer é desativado e os três fragmentos contidos na memória do receptor são descartados. Logo depois, chega o último fragmento. O que deve ser feito com ele?

38. No IP, o checksum abrange apenas o cabeçalho, não os dados. Para você, por que essa estrutura foi escolhida?

39. Uma pessoa que mora em Boston viaja para Minneapolis levando consigo seu computador portátil. Para sua surpresa, a LAN em Minneapolis é do tipo IP sem fio; portanto, ela não precisa se conectar. Mesmo assim, será preciso percorrer toda a empresa com agentes locais e externos para fazer com que as mensagens de correio eletrônico e outros tipos de tráfego cheguem corretamente?

40. O IPv6 utiliza endereços de 16 bytes. Se um bloco de 1 milhão de endereços for alocado a cada picossegundo, qual será a duração desses endereços?

41. O campo *Protocolo* usado no cabeçalho do IPv4 não é encontrado no cabeçalho fixo do IPv6. Por quê?

42. Quando o protocolo IPv6 for introduzido, o protocolo ARP tem de ser alterado? Se houver necessidade de mudanças, elas serão conceituais ou técnicas?

43. Crie um programa para simular o roteamento usando o algoritmo de flooding. Cada pacote deve conter um contador que é decrementado a cada hop. Quando o contador chegar a zero, o pacote será descartado. O tempo é discreto, com cada interface tratando de um pacote a cada intervalo. Crie três versões do programa: todas as interfaces sofrem flooding, todas as interfaces com exceção da de entrada sofrem flooding, e somente as k melhores interfaces (escolhidas estatisticamente) sofrem flooding. Compare o algoritmo de flooding com o roteamento determinístico ($k = 1$) em termos de atraso e de largura de banda utilizada.

44. Crie um programa capaz de simular uma rede de computadores usando tempo discreto. O primeiro pacote de cada fila do roteador cria um hop por intervalo. Cada roteador tem apenas um número finito de buffers. Se um pacote chegar e não houver espaço, ele será descartado e não será retransmitido. Em vez disso, há um protocolo ponto a ponto completo, com intervalos de timeout e pacotes de confirmação, que em algum momento regenera o pacote do roteador de origem. Represente o throughput da rede como uma função do intervalo de timeout ponto a ponto, parametrizado pela taxa de erros.

45. Crie uma função para realizar o encaminhamento em um roteador IP. O procedimento tem um único parâmetro, um endereço IP. Ele também tem acesso a uma tabela global que consiste em um array de triplas. Cada tripla contém três inteiros: um endereço IP, uma máscara de sub-

-rede e a interface de saída a ser usada. A função pesquisa o endereço IP na tabela usando o CIDR e retorna como seu valor a interface que deverá ser usada.

46. Use o programa *traceroute* (UNIX) ou o programa *tracert* (Windows) para traçar a rota desde seu computador até várias universidades em outros continentes. Faça uma lista dos enlaces transoceânicos que você descobrir. Alguns sites que você deve tentar são:

www.berkeley.edu (Califórnia)
www.mit.edu (Massachusetts)
www.vu.nl (Amsterdã)
www.ucl.ac.uk (Londres)
www.usyd.edu.au (Sydney)
www.u-tokyo.ac.jp (Tóquio)
www.uct.ac.za (Cidade do Cabo)

A camada de transporte

Capítulo 6

Com a camada de rede, a camada de transporte é o núcleo da hierarquia de protocolos. A camada de rede oferece remessa de pacotes fim a fim usando datagramas ou circuitos virtuais. A camada de transporte se baseia na camada de rede para oferecer transporte de dados de um processo em uma máquina de origem a um processo em uma máquina de destino com um nível de confiabilidade desejado independentemente das redes físicas em uso no momento. Ela provê as abstrações de que as aplicações precisam para usar a rede. Sem a camada de transporte, todo o conceito de protocolos em camadas faria pouco sentido. Neste capítulo estudaremos em detalhes a camada de transporte, bem como seus serviços e a seleção de projeto de uma API para enfrentar questões de confiabilidade, conexões e controle de congestionamento, protocolos como TCP e UDP, e desempenho.

6.1 O serviço de transporte

Nas próximas seções, apresentaremos uma introdução ao serviço de transporte. Veremos que tipo de serviço é oferecido à camada de aplicação. Para tornar mais concreta a questão do serviço de transporte, examinaremos dois conjuntos de primitivas da camada de transporte. Primeiro, estudaremos uma primitiva simples (mas hipotética) para mostrar as ideias básicas. Em seguida, analisaremos a interface comumente utilizada na Internet.

6.1.1 Serviços oferecidos às camadas superiores

O principal objetivo da camada de transporte é oferecer um serviço confiável, eficiente e econômico a seus usuários, que, em geral, são processos presentes na camada de aplicação. Para atingir esse objetivo, a camada de transporte utiliza vários serviços oferecidos pela camada de rede. O hardware/software da camada de transporte que executa o trabalho é chamado **entidade de transporte**. A entidade de transporte pode estar localizada no núcleo do sistema operacional, em um pacote da biblioteca vinculada às aplicações em rede, em outro processo do usuário ou na placa de interface de rede. As duas primeiras opções são mais comuns na Internet. O relacionamento (lógico) existente entre as camadas de rede, de transporte e de aplicação está ilustrado na Figura 6.1.

Assim como existem dois tipos de serviço de rede — o serviço orientado a conexões e o serviço não orientado a conexões —, também existem dois tipos de serviço de transporte. O serviço de transporte orientado a conexões é semelhante ao serviço de rede orientado a conexões em muitos aspectos. Em ambos os casos, as conexões têm três fases: o estabelecimento, a transferência de dados e o encerramento. O endereçamento e o controle de fluxo também são semelhantes em ambas as camadas. Além disso, o serviço de transporte não orientado a conexões é semelhante ao serviço de rede não orientado a conexões. Porém, observe que pode ser difícil oferecer um serviço de

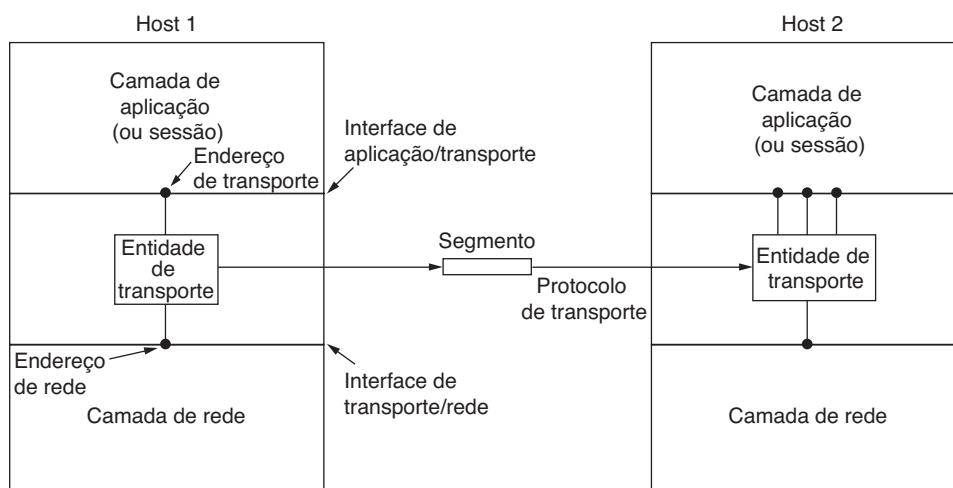



Figura 6.1 As camadas de rede, de transporte e de aplicação.

transporte não orientado a conexões sobre um serviço de rede orientado a conexões, pois é ineficaz estabelecer uma conexão para enviar um único pacote e encerrá-la imediatamente depois.

Diante disso, fazemos a seguinte pergunta: se o serviço da camada de transporte é tão semelhante ao serviço da camada de rede, por que há duas camadas distintas? Por que uma só camada não é suficiente? A resposta, embora sutil, é de importância crucial. O código de transporte funciona inteiramente nas máquinas dos usuários, mas a camada de rede funciona principalmente nos roteadores, cuja operação é de responsabilidade da concessionária de comunicações (pelo menos no caso de uma rede geograficamente distribuída). O que acontecerá se a camada de rede oferecer um serviço inadequado? E se perder pacotes com frequência? O que acontecerá se os roteadores apresentarem falhas ocasionais?

Problemas acontecem. Os usuários não têm nenhum controle real sobre a camada de rede, portanto, não podem resolver o problema de um serviço ineficaz usando roteadores melhores ou aumentando o controle de erros na camada de enlace de dados, pois não possuem roteadores. A única possibilidade é colocar sobre a camada de rede outra camada que melhore a qualidade do serviço. Se, em uma rede não orientada a conexões, pacotes forem perdidos ou danificados, a entidade de transporte pode detectar o problema e compensá-lo usando retransmissões. Se, em uma rede orientada a conexões, uma entidade de transporte for informada no meio de uma longa transmissão de que sua conexão de rede foi encerrada de forma abrupta, sem nenhuma indicação do que ocorreu com os dados que estavam sendo transferidos, ela poderá gerar uma nova conexão de rede para a entidade de transporte remota. Usando essa nova conexão de rede, ela poderá enviar uma consulta à entidade remota para saber quais dados chegaram e quais não chegaram, e depois retomar a transmissão a partir da interrupção.

Em síntese, a existência de uma camada de transporte torna o serviço de transporte mais confiável que o serviço de rede subjacente. Além disso, as primitivas de serviço de transporte podem ser implementadas sob a forma de chamadas a procedimentos de biblioteca, a fim de torná-las independentes das primitivas de serviço de rede. As chamadas do serviço de rede podem variar muito de uma rede para outra (por exemplo, chamadas baseadas em uma Ethernet não orientada a conexões podem ser muito diferentes das chamadas em uma rede WiMAX orientada a conexões). Ocultar o serviço de rede por trás de um conjunto de primitivas de serviço de transporte garante que a mudança da rede simplesmente requer substituição de um conjunto de procedimentos de biblioteca por outro que faça a mesma coisa com um serviço subjacente diferente.

Graças à camada de transporte, os desenvolvedores de aplicações distribuídas em rede podem escrever códigos de acordo com um conjunto de primitivas-padrão e permitir que esses programas funcionem em uma grande variedade de redes, sem a preocupação de lidar com diferentes interfaces de redes e níveis de confiabilidade. Se todas as redes reais fossem perfeitas e se todas tivessem as mesmas primitivas de serviço, e ainda a garantia de nunca mudar, provavelmente a camada de transporte não seria necessária. Porém, na vida real ela cumpre a função primordial de tornar as camadas superiores do projeto imunes à tecnologia, ao projeto e a imperfeições da rede.

Por essa razão, muitas pessoas fazem distinção entre as camadas de 1 a 4, por um lado, e as camadas acima de 4, por outro. As quatro primeiras camadas são consideradas o **provedor de serviços de transporte**, enquanto as camadas superiores constituem o **usuário de serviços de transporte**. Essa distinção entre provedor e usuário tem um impacto considerável sobre o projeto das camadas e coloca a camada de transporte em uma posição-chave, pois ela representa a principal fronteira entre o provedor e o usuário do serviço de transmissão de dados confiável. Esse é o nível que as aplicações veem.

6.1.2 Primitivas do serviço de transporte

Para permitir que os usuários tenham acesso ao serviço de transporte, a camada de transporte deve oferecer algumas operações aos aplicativos em rede, ou seja, uma interface de serviço de transporte. Cada serviço de transporte tem sua própria interface. Nesta seção, veremos primeiro um serviço de transporte (hipotético) simples e sua interface, a fim de examinarmos os fundamentos desse tipo de serviço. Na próxima seção, estudaremos um exemplo real.

O serviço de transporte é semelhante ao serviço de rede, mas existem algumas diferenças importantes entre eles. A principal delas é que o serviço de rede representa o modelo oferecido pelas redes reais, com todos os atributos. As redes reais podem perder pacotes; portanto, o serviço de rede, em geral, não é confiável.

Em contraste, o serviço de transporte orientado a conexões é confiável. É óbvio que as redes reais não são infalíveis, mas essa é exatamente a função da camada de transporte — oferecer um serviço confiável sobre uma rede não confiável.

Como exemplo, considere dois processos em uma única máquina, conectados por canais (pipes) no UNIX (ou qualquer outro serviço de comunicação entre processos). Supomos que a conexão entre eles é perfeita, sem levar em consideração confirmações, pacotes perdidos, congestionamentos etc. Esses processos esperam ter uma conexão 100 por cento confiável. O processo *A* insere os dados em uma extremidade do canal, e o processo *B* os recebe na outra. O serviço de transporte orientado a conexões consiste em ocultar as imperfeições do serviço de rede, de modo que os processos do usuário possam simplesmente supor a existência de um fluxo de bits livre de erros mesmo quando estão em máquinas diferentes.

A propósito, a camada de transporte também pode oferecer um serviço não confiável (de datagramas). Porém, há relativamente pouco a dizer sobre esse assunto e, assim, neste capítulo vamos nos concentrar no serviço de transporte orientado a conexões. Apesar disso, há algumas aplicações, como a computação cliente-servidor e o streaming de multimídia, que se beneficiam do serviço de transporte não orientado a conexões; por essa razão, estudaremos alguns detalhes sobre ele mais adiante.

A segunda diferença entre o serviço de rede e o serviço de transporte está relacionada ao destinatário do serviço. O serviço de rede só é usado pelas entidades de transporte. Poucos usuários criam suas próprias entidades de transporte; portanto, poucos usuários ou programas veem a estrutura do serviço de rede. Por outro lado, muitos programas (e programadores) veem as primitivas de transporte. Por isso, o serviço de transporte deve ser adequado e fácil de usar.

Para ter uma ideia de como deve ser um serviço de transporte, considere as cinco primitivas apresentadas na Tabela 6.1. Essa interface de transporte é uma estrutura bem básica, mas denota o que uma interface de transporte orientada a conexões deve fazer. Ela permite aos programas aplicativos estabelecer, usar e encerrar uma conexão, o que é suficiente para muitas aplicações.

Primitiva	Pacote enviado	Significado
LISTEN	(nenhum)	Bloqueia até algum processo tentar conectar
CONNECT	CONNECTION REQ.	Tenta ativamente estabelecer uma conexão
SEND	DATA	Envia informação
RECEIVE	(nenhum)	Bloqueia até que um pacote de dados chegue
DISCONNECT	DISCONNECTION REQ.	Solicita uma liberação da conexão

Tabela 6.1 | As primitivas para um serviço de transporte simples.

Para entender como essas primitivas devem ser usadas, considere uma aplicação com um servidor e uma série de clientes remotos. Primeiro, o servidor executa uma primitiva LISTEN, geralmente chamando um procedimento de biblioteca que emite uma chamada de sistema para bloquear o servidor até que um cliente apareça. Quando um cliente quer se comunicar com o servidor, ele executa uma primitiva CONNECT. Para executar essa primitiva, a entidade de transporte bloqueia o transmissor e envia um pacote ao servidor. Encapsulada no campo carga útil desse pacote encontra-se uma mensagem da camada de transporte destinada à entidade de transporte do servidor.

Faremos agora algumas observações rápidas sobre a terminologia. Por falta de um termo mais adequado, usaremos **segmento** para denominar as mensagens enviadas de uma entidade de transporte a outra entidade de transporte. TCP, UDP e outros protocolos da Internet utilizam esse termo. Alguns protocolos mais antigos usavam o desajeitado acrônimo **TPDU** (unidade de dados de protocolo de transporte, do inglês **Transport Protocol Data Unit**). Esse termo não é mais muito usado, mas você poderá encontrá-lo em artigos e livros mais antigos.

Portanto, os segmentos (intercambiados pela camada de transporte) estão contidos em pacotes (intercambiados pela camada de rede). Por sua vez, os pacotes estão contidos em quadros (intercambiados pela camada de enlace de dados). Quando um quadro chega, a camada de enlace de dados processa o cabeçalho do quadro e, se o endereço de destino combina para a entrega local, transmite o conteúdo do campo de carga útil do quadro à entidade de rede. Em seguida, a entidade de rede processa o cabeçalho do pacote e envia o conteúdo do campo carga útil do pacote à entidade de transporte. Esse aninhamento é ilustrado na Figura 6.2.

Voltando ao exemplo de cliente-servidor, a chamada CONNECT do cliente envia um segmento CONNECTION REQUEST ao servidor. Quando o segmento chega, a entidade de transporte verifica se o servidor está bloqueado em uma primitiva LISTEN (ou seja, apto a administrar solicitações). Em

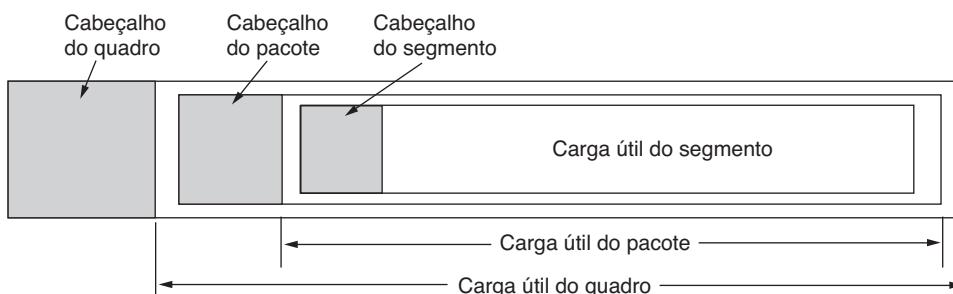



Figura 6.2 | Aninhamento de TPDUs, pacotes e quadros.

seguida, ele desbloqueia o servidor e transmite um segmento CONNECTION ACCEPTED de volta para o cliente. Quando esse segmento chega a seu destino, o cliente é desbloqueado e a conexão é estabelecida.

A partir desse momento é possível intercambiar dados utilizando-se as primitivas SEND e RECEIVE. Em sua forma mais simples, qualquer uma das partes pode executar uma primitiva RECEIVE (bloqueio) para aguardar que a outra execute uma primitiva SEND. Quando o segmento chega a seu destino, o receptor é desbloqueado. Em seguida, ele pode processar o segmento e enviar uma resposta. Esse sistema funciona bem desde que as partes controlem de quem é a vez de enviar os dados.

Cabe observar que, na camada de transporte, até mesmo uma troca de dados simples é unidirecional e mais complexa do que na camada de rede. Cada pacote de dados enviado também será confirmado (ao final). Os pacotes que transportam segmentos de controle também são confirmados, de forma explícita ou implícita. Essas confirmações são gerenciadas por entidades de transporte que utilizam o protocolo da camada de rede e não são visíveis para os usuários de transporte. Da mesma forma, as entidades de transporte têm de lidar com timers e retransmissões. Esse mecanismo também não é percebido pelos usuários de transporte. Para eles, uma conexão é um canal de bits confiável: um usuário envia bits por uma das extremidades e eles são recebidos na outra como em um passe de mágica. A habilidade de ocultar sua complexidade é o que torna os protocolos em camadas uma ferramenta tão útil.

Quando uma conexão não é mais necessária, ela deve ser encerrada para desocupar espaço na tabela alocada dentro das duas entidades de transporte. O encerramento da conexão tem duas variantes: assimétrica e simétrica. Na variante assimétrica, qualquer um dos usuários de transporte pode emitir uma primitiva DISCONNECT, fazendo com que um segmento DISCONNECT seja enviado à entidade de transporte remota. Quando esse segmento chega a seu destino, a conexão é encerrada.

Na variante simétrica, cada direção da comunicação é encerrada separadamente e de forma independente da outra. Quando uma das partes executa uma primitiva DISCONNECT, isso significa que não há mais dados a enviar, mas ainda é possível receber dados enviados pela outra parte. Nesse modelo, a conexão só é encerrada quando os dois lados tiverem executado uma primitiva DISCONNECT.

Um diagrama de estados para o estabelecimento e encerramento de uma conexão com essas primitivas simples é mostrado na Figura 6.3. Cada transição é acionada por algum evento, seja ele uma primitiva executada pelo usuário de transporte local ou um pacote que chega. Por simplicidade, supomos que cada segmento é confirmado separadamente e que um modelo de desconexão simétrica está sendo usado, com o cliente dando início ao procedimento. Cabe observar que esse modelo é muito simples. Veremos adiante outros modelos mais realistas, quando descrevermos o modo como o TCP funciona.

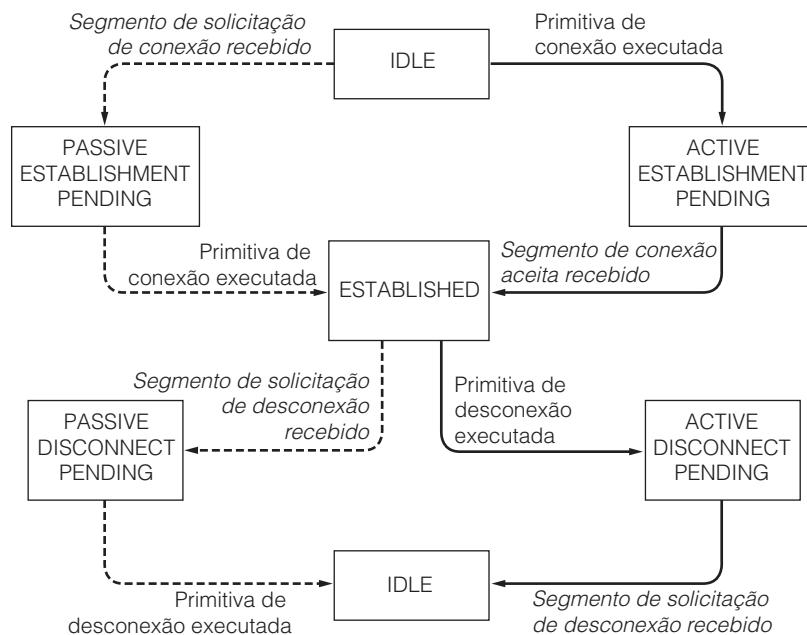



Figura 6.3 | Diagrama de estados para um esquema simples de gerenciamento de conexão. As transições identificadas em itálico são causadas pela chegada de pacotes. As linhas contínuas mostram a sequência de estados do cliente. As linhas tracejadas mostram a sequência de estados do servidor.

6.1.3 Soquetes de Berkeley

Agora, vamos analisar resumidamente outro conjunto de primitivas de transporte, as primitivas de soquetes de programação usadas para o TCP. Os soquetes foram lançados inicialmente como parte da distribuição do software UNIX 4.2BSD de Berkeley em 1983. Eles logo se tornaram populares. As primitivas agora são bastante usadas para programação na Internet em muitos sistemas operacionais, especialmente sistemas baseados no UNIX, e existe uma API em estilo soquete para o Windows, chamada 'winsock'.

Essas primitivas são mostradas na Tabela 6.2. *Grosso modo*, elas seguem o mesmo modelo do nosso primeiro exemplo, mas oferecem mais recursos e maior flexibilidade. Não vamos mostrar os segmentos correspondentes a elas nesta seção; essa discussão ficará para mais adiante.

As quatro primeiras primitivas na lista são executadas pelos servidores nessa mesma ordem. A primitiva SOCKET cria um novo ponto final e aloca espaço da tabela para ele na entidade de transporte. Os parâmetros da chamada especificam o formato de endereçamento a ser usado, o tipo de serviço desejado (por exemplo, um fluxo de bytes confiável) e o protocolo. Uma chamada SOCKET bem-sucedida retorna um descritor de arquivo comum que será usado nas chamadas subsequentes, exatamente como uma chamada OPEN sobre um arquivo.

Os soquetes recém-criados não têm endereços de rede; esses endereços são atribuídos através da primitiva BIND. Uma vez que um servidor tenha designado um endereço para um soquete, os clientes remotos já podem se conectar a ele. A razão para uma chamada SOCKET não criar um endereço diretamente é que alguns processos consideram importante manter seus endereços (por exemplo, eles vêm usando o mesmo endereço há muitos anos e todos já o conhecem), ao passo que outros não se importam com isso.

Em seguida, temos a chamada LISTEN, que aloca espaço para a fila de chamadas recebidas, caso vários clientes tentem se conectar ao mesmo tempo. Ao contrário da chamada LISTEN em nosso primeiro exemplo, no modelo de soquetes, LISTEN não é uma chamada de bloqueio.

Primitiva	Significado
SOCKET	Criar um novo ponto final de comunicação
BIND	Anexar um endereço local a um soquete
LISTEN	Anunciar a disposição para aceitar conexões; mostrar o tamanho da fila
ACCEPT	Bloquear o responsável pela chamada até uma tentativa de conexão ser recebida
CONNECT	Tentar estabelecer uma conexão ativamente
SEND	Enviar alguns dados através da conexão
RECEIVE	Receber alguns dados da conexão
CLOSE	Encerrar a conexão

Tabela 6.2 | As primitivas de soquetes para TCP.

Para bloquear a espera por uma conexão de entrada, o servidor executa uma primitiva ACCEPT. Quando chega um segmento solicitando uma conexão, a entidade de transporte cria um novo soquete com as mesmas propriedades do original e retorna um descritor de arquivo para ele. Em seguida, o servidor pode desviar um processo ou uma thread para tratar a conexão no novo soquete e voltar a esperar pela próxima conexão no soquete original. ACCEPT retorna um descritor de arquivo normal, que pode ser usado para ler e gravar da maneira padrão, como no caso de arquivos.

Agora, vamos ver o que acontece no lado cliente. Aqui também é preciso criar primeiro um soquete usando a primitiva SOCKET, mas a primitiva BIND não é necessária, pois o endereço usado não é importante para o servidor. A primitiva CONNECT bloqueia o responsável pela chamada e inicia o processo de conexão. Quando a conexão é concluída (ou seja, quando o segmento apropriado é recebido do servidor), o processo cliente é desbloqueado e a conexão é estabelecida. Depois disso, os dois lados podem usar as primitivas SEND e RECEIVE para transmitir e receber dados através da conexão full-duplex. As chamadas do sistema READ e WRITE padrão do UNIX também podem ser usadas, se nenhuma das opções especiais de SEND e RECEIVE for necessária.

O encerramento da conexão com soquetes é simétrico. Quando ambos os lados tiverem executado uma primitiva CLOSE, a conexão será encerrada.

Soquetes têm provado ser extremamente populares e são os padrões de fato para abstrair serviços de transporte às aplicações. A API de soquete normalmente é usada com o protocolo TCP para oferecer um serviço orientado à conexão, chamado **fluxo de bytes confiável**, que é simplesmente um canal de bits confiável, que já descrevemos. Contudo, outros protocolos poderiam ser usados para a implementação desse serviço usando a mesma API. Tudo deverá ser o mesmo para os usuários do serviço de transporte.

Um ponto forte da API de soquetes é que ela pode ser usada por uma aplicação para outros serviços de transporte. Por exemplo, os soquetes podem ser usados com um serviço de transporte não orientado a conexões. Nesse caso, CONNECT define o endereço do peer de transporte remoto e SEND e RECEIVE enviam e recebem datagramas de e para o peer remoto. (Também é comum usar um conjunto expandido de chamadas, por exemplo, SENDTO e RECEIVEFROM, que enfatizam mensagens e não limitam uma aplicação a um único peer de transporte.) Os soquetes também podem ser usados com protocolos de transporte que oferecem um fluxo de mensagens, em vez de um fluxo de bytes, e que realizam ou não controle de congestionamento. Por exemplo, **DCCP** (do inglês **Datagram Congestion Controlled Protocol**) é uma versão do UDP com

controle de congestionamento (Kohler et al., 2006). Fica a critério dos usuários de transporte entender qual serviço eles estão obtendo.

Entretanto, os soquetes provavelmente não são a última palavra em interfaces de transporte. Por exemplo, as aplicações normalmente trabalham com um grupo de fluxos relacionados, como um navegador Web que solicita vários objetos do mesmo servidor. Com os soquetes, o mais natural é que os programas aplicativos usem um fluxo por objeto. Essa estrutura significa que o controle de congestionamento é aplicado separadamente para cada fluxo, e não para todo o grupo, o que não é o ideal. Fica para a aplicação o peso de gerenciar o conjunto. Protocolos e interfaces mais novas têm sido criados para dar suporte a grupos de fluxos relacionais com mais eficiência e simplicidade para a aplicação. Dois exemplos são **SCTP** (***S*tream Control Transmission Protocol**), definido na RFC 4960, e **SST** (**Structured Stream Transport**) (Ford, 2007). Esses protocolos precisam mudar ligeiramente a API de soquetes para que obtenham os benefícios de grupos de fluxos relacionados, e eles também dão suporte a recursos como uma mistura de tráfego orientado a conexões e não orientado a conexões, e até mesmo múltiplos caminhos pela rede. O tempo dirá se eles terão sucesso.

6.1.4 Exemplo de programação com soquetes: um servidor de arquivos da Internet

Como um exemplo de utilização das chamadas por soquetes, considere o código do cliente e o código do servidor do Quadro 6.1. Nesse quadro, temos um servidor de arquivos da Internet muito primitivo, juntamente com um exemplo de cliente que o utiliza. O código tem muitas limitações (descritas a seguir), mas, em princípio, o código do servidor pode ser compilado e executado em qualquer sistema UNIX conectado à Internet. O código do cliente pode então ser compilado e executado em qualquer outra máquina UNIX conectada à Internet, em qualquer lugar do mundo. O código do cliente pode ser executado com parâmetros apropriados para buscar qualquer arquivo ao qual o servidor tem acesso em sua máquina. O arquivo é gravado na saída-padrão, que, é claro, pode ser redirecionada para um arquivo ou um canal.

Vamos examinar primeiro o código do servidor. Ele começa incluindo alguns cabeçalhos-padrão, e os três últimos cabeçalhos contêm as principais definições e estruturas de dados relacionadas à Internet. Em seguida, temos uma definição de *SERVER_PORT* como 12345. Esse número foi escolhido arbitrariamente. Qualquer número entre 1.024 e 65.535 também funcionará, desde que não esteja em uso por algum outro processo; as portas abaixo de 1.023 são reservadas para usuários privilegiados.

As duas linhas seguintes no servidor definem duas constantes necessárias. A primeira define, em bytes, o tamanho do bloco usado na transferência de arquivos. A segunda determina quantas conexões pendentes podem ser mantidas antes de ser descartadas conexões adicionais que chegarem.

Depois das declarações de variáveis locais, tem início o código do servidor. Ele começa iniciando uma estrutura de dados que conterá o endereço IP do servidor. Essa estrutura de dados logo será vinculada ao soquete do servidor. A chamada a *memset* define a estrutura de dados com somente valores 0. As três atribuições seguintes preenchem três de seus campos. A última dessas atribuições contém a porta do servidor. As funções *htonl* e *htons* estão relacionadas com a conversão de valores para um formato-padrão, de modo que o código possa funcionar corretamente em máquinas little-endian (por exemplo, Intel x86) e em máquinas big-endian (por exemplo, SPARC). Sua semântica exata não é relevante no momento.

Em seguida, o servidor cria um soquete e verifica se ele contém erros (indicados por $s < 0$). Em uma versão de produção do código, a mensagem de erro poderia ser um pouco mais explicativa. A chamada a *setsockopt* é necessária para permitir que a porta seja reutilizada, de forma que o servidor possa funcionar indefinidamente, recebendo solicitação após solicitação. Agora, o endereço IP é vinculado ao soquete e é realizada uma verificação para saber se a chamada a *bind* teve sucesso. A última etapa da inicialização é a chamada a *listen*, para anunciar a disposição do servidor para aceitar chamadas de entrada e informar ao sistema que ele deve armazenar um número de chamadas igual a *QUEUE_SIZE*, no caso de chegarem novas solicitações enquanto o servidor ainda estiver processando a solicitação atual. Se a fila estiver cheia e chegarem outras solicitações, elas serão descartadas de forma transparente.

Nesse momento, o servidor entra em seu loop principal, do qual ele nunca sai. A única maneira de interromper o loop é encerrá-lo externamente. A chamada a *accept* bloqueia o servidor até algum cliente tentar estabelecer uma conexão com ele. Se a chamada a *accept* tiver êxito, ela retornará um descritor de soquete que poderá ser usado para leitura e gravação, de maneira análoga ao uso de descritores de arquivos em operações de leitura e gravação em canais de mídia. Porém, diferentemente dos canais de mídia, que são unidirecionais, os soquetes são bidirecionais, e assim o soquete aceito (*sa*) pode ser usado para realizar a leitura da conexão e também para efetuar gravações. Um descritor de arquivo em um canal de mídia serve para leitura ou gravação, mas não ambos.

Depois que a conexão é estabelecida, o servidor lê o nome do arquivo. Se o nome ainda não estiver disponível, o servidor será bloqueado esperando por ele. Após obter o nome do arquivo, o servidor o abre e, em seguida, entra em um loop que, alternadamente, lê blocos do arquivo e os grava no soquete, até copiar o arquivo inteiro. Depois, o servidor fecha o arquivo e a conexão, e espera que a próxima conexão apareça. Ele repete esse loop indefinidamente.

```
/* Esta página contém um programa cliente que pode solicitar um arquivo do*/
/* programa servidor na próxima página. O servidor responde enviando o arquivo inteiro.*/


#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define SERVER_PORT 12345                /* arbitrário, mas cliente e servidor devem combinar */
#define BUF_SIZE 4096                    /* tamanho do bloco de transferência */

int main(int argc, char **argv)
{
  int c, s, bytes;
  char buf[BUF_SIZE];                    /* buffer para arquivo de entrada */
  struct hostent *h;                     /* informações sobre servidor */
  struct sockaddr_in channel;            /* mantém endereço IP */

  if (argc != 3) fatal("Usage: client server-name file-name");
  h = gethostbyname(argv[1]);            /* pesquisa endereço IP do host */
  if (!h) fatal("gethostbyname failed");

  s = socket(PF_INET,SOCK_STREAM,IPPROTO_TCP);
  if (s <0) fatal("socket");
  memset(&channel, 0, sizeof(channel));
  channel.sin_family= AF_INET;
  memcpy(&channel.sin_addr.s_addr, h->h_addr,h->h_length);
  channel.sin_port= htons(SERVER_PORT);

  c = connect(s, (struct sockaddr *) &channel, sizeof(channel));
  if (c < 0) fatal("connect failed");

  /* Conexão agora estabelecida. Envia nome do arquivo com byte 0 no final. */
  write(s, argv[2], strlen(argv[2])+1);

  /* Captura o arquivo e o escreve na saída padrão. */
  while (1) {
    bytes = read(s, buf, BUF_SIZE);      /* lê do soquete */
    if (bytes <= 0) exit(0);             /* verifica final de arquivo */
    write(1, buf, bytes);                /* escreve na saída padrão */
  }
}

fatal(char *string)
{
  printf("%s\n", string);
  exit(1);
}
```

Quadro 6.1 | Código do cliente usando soquetes. O código do servidor está na página seguinte.

```
#include <sys/types.h>                          /* Este é o código do servidor */
#include <sys/fcntl.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define SERVER_PORT 12345                       /* arbitrário, mas cliente e servidor devem combinar */

#define BUF_SIZE 4096                           /* tamanho do bloco de transferência */
#define QUEUE_SIZE 10

int main(int argc, char *argv[])
{
 int s, b, l, fd, sa, bytes, on = 1;
 char buf[BUF_SIZE];                            /* buffer para arquivo de saída */
 struct sockaddr_in channel;                    /* mantém endereço IP */

 /* Monta estrutura de endereços para vincular ao soquete. */
 memset(&channel, 0, sizeof(channel));          /* canal zero */
 channel.sin_family = AF_INET;
 channel.sin_addr.s_addr = htonl(INADDR_ANY);
 channel.sin_port = htons(SERVER_PORT);

 /* Abertura passiva. Espera a conexão. */
 s = socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);   /* cria soquete */
 if (s < 0) fatal("socket failed");
 setsockopt(s, SOL_SOCKET,SO_REUSEADDR,(char *) &on, sizeof(on));

 b = bind(s, (struct sockaddr *) &channel, sizeof(channel));
 if (b < 0) fatal("bind failed");

 l = listen(s, QUEUE_SIZE);                     /* especifica tamanho da fila */
 if (l < 0) fatal("listen failed");

 /* O soquete agora está preparado e vinculado. Espera conexão e a processa. */
 while (1) {
  sa = accept(s, 0, 0);                         /* bloqueia solicitação de conexão */
  if (sa < 0) fatal("accept failed");

  read(sa, buf, BUF_SIZE);                      /* lê nome do arquivo do soquete */

  /* Captura e retorna o arquivo. */
  fd = open(buf, O_RDONLY);                     /* abre arquivo para ser enviado de volta */
  if (fd < 0) fatal("open failed");

  while (1) {
   bytes = read(fd, buf, BUF_SIZE);             /* lê do arquivo */
   if (bytes <= 0) break;                       /* verifica se é final do arquivo */
   write(sa, buf, bytes);                       /* grava bytes no soquete */
  }
  close(fd);                                    /* fecha arquivo */
  close(sa);                                    /* fecha conexão */
 }
}
```

Agora vamos examinar o lado do código do cliente. Para entender como ele funciona, é necessário compreender como é chamado. Supondo-se que ele seja denominado *client*, uma chamada típica será:

client flits.cs.vu.nl /usr/tom/nomearq >f

Essa chamada só funciona se o servidor já estiver em execução em *flits.cs.vu.nl* e o arquivo */usr/tom/nomearq* existir, e se o servidor tiver acesso de leitura para o arquivo. Se a chamada for bem-sucedida, o arquivo será transferido pela Internet e gravado em *f*; depois disso, o programa cliente será encerrado. Tendo em vista que o processo servidor continua após uma transferência, o cliente pode ser iniciado repetidamente para obter outros arquivos.

O código do cliente começa com algumas inclusões e declarações. A execução inicia verificando se o código foi chamado com o número correto de argumentos (*argc* = 3 representa o nome do programa e mais dois argumentos). Observe que *argv* [1] contém o nome do servidor (por exemplo, *flits.cs.vu.nl*) e é convertido em um endereço IP por *get*host*byname*. Essa função utiliza o DNS para pesquisar o nome. Estudaremos o DNS no Capítulo 7.

Em seguida, um soquete é criado e iniciado. Depois disso, o cliente tenta estabelecer uma conexão TCP para o servidor, usando *connect*. Se o servidor estiver funcionando na máquina nomeada e conectado a *SERVER_PORT*, e se ele estiver ocioso ou tiver espaço em sua fila *listen*, a conexão será (em algum momento) estabelecida. Usando a conexão, o cliente envia o nome do arquivo gravando-o no soquete. O número de bytes enviados é uma unidade maior que o nome em si, tendo em vista que o byte 0 que encerra o nome também deve ser enviado para informar ao servidor em que ponto o nome termina.

Agora o cliente entra em um loop, lendo o arquivo bloco por bloco do soquete e copiando-o na saída-padrão. Ao terminar, ele simplesmente encerra o loop.

A função *fatal* imprime uma mensagem de erro e se encerra. O processo servidor precisa do mesmo procedimento, mas ele foi omitido por falta de espaço na página. Tendo em vista que o cliente e o servidor são compilados separadamente e em geral funcionam em computadores diferentes, eles não podem compartilhar o código da função *fatal*.

Apenas para registrar, esse servidor não é a última palavra em desempenho. Sua verificação de erros é escassa e seu relatório de erros é medíocre. Como ele maneja todas as demandas de modo estritamente sequencial (porque tem apenas um thread), seu desempenho é insatisfatório. É claro que ele nunca ouviu falar de segurança, e usar chamadas comuns do sistema UNIX não é a última palavra em independência de plataforma. Ele também faz algumas suposições tecnicamente ilegais, como a de considerar que o nome do arquivo cabe no buffer e é transmitido de forma automática. Apesar dessas limitações, é um servidor de arquivos da Internet completo e funcional. Nos exercícios, o leitor será convidado a aperfeiçoá-lo. Para obter mais informações sobre programação com soquetes, consulte Donahoo e Calvert (2008, 2009).

6.2 Elementos dos protocolos de transporte

O serviço de transporte é implementado por um **protocolo de transporte** usado entre duas entidades de transporte. Em alguns aspectos, esses protocolos lembram os de enlace de dados que estudamos em detalhes no Capítulo 3. Ambos têm de lidar com o controle de erros, com a definição de sequências e com o controle de fluxo, entre outros itens.

Entretanto, existem diferenças significativas entre os dois. Essas diferenças ocorrem devido às peculiaridades dos ambientes nos quais os dois protocolos operam, como mostra a Figura 6.4. Na camada de enlace de dados, dois roteadores se comunicam diretamente através de um canal físico com ou sem conexão por cabo, enquanto na camada de transporte esse canal físico é substituído pela rede inteira. Essa diferença tem muitas implicações importantes para os protocolos.

Por um lado, na camada de enlace de dados o roteador não precisa especificar com qual roteador deseja se comunicar, pois cada linha de saída especifica de modo exclusivo determinado roteador. Na camada de transporte, é necessário o endereçamento explícito de destinos.

Por outro, o processo de estabelecimento de uma conexão através de um cabo, como mostra a Figura 6.4(a), é simples: a outra extremidade está sempre presente (a me-

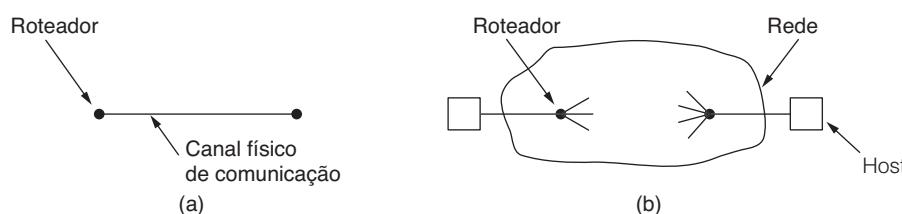



Figura 6.4 | (a) Ambiente da camada de enlace de dados. (b) Ambiente da camada de transporte.

nos que haja alguma falha). De qualquer modo, não há muito a fazer. Até mesmo em enlaces sem fios, o processo não é muito diferente. Apenas enviar uma mensagem já é suficiente para que ela alcance todos os outros destinos. Se a mensagem não for confirmada devido a um erro, ela pode ser reenviada. Já na camada de transporte, o estabelecimento inicial da conexão é mais complicado, como veremos mais adiante.

Outra diferença (extremamente inoportuna) entre a camada de enlace de dados e a camada de transporte é a possível existência de capacidade de armazenamento na rede. Quando um roteador envia um pacote por um enlace, ele pode chegar a seu destino ou se perder, mas não ricocheteia nem se esconde em algum canto para emergir de repente, em um momento inoportuno, depois que outros pacotes foram enviados, muito tempo depois. Se a rede utilizar datagramas, que são roteados independentemente no interior da rede, haverá uma probabilidade significativa de que um pacote possa ficar armazenado por alguns segundos e ser entregue mais tarde, fora da ordem esperada. As consequências da habilidade de atrasar e duplicar pacotes podem às vezes ser desastrosas e exigir o uso de protocolos especiais para transportar a informação corretamente.

A última diferença entre as camadas de enlace de dados e de transporte é mais uma questão de grau, e não de espécie. Os controles de buffers e de fluxo são necessários em ambas as camadas, mas a presença de um número grande e dinamicamente variável de conexões na camada de transporte pode exigir outra estratégia que não a da camada de enlace de dados. No Capítulo 3, vimos que alguns dos protocolos alocam um número fixo de buffers para cada linha. Portanto, quando um quadro chegar, sempre haverá um buffer disponível. Na camada de transporte, o grande número de conexões que precisam ser gerenciadas e as variações na largura de banda que cada conexão pode receber tornam menos atraente a ideia de dedicar vários buffers a cada uma. Nas próximas seções, vamos analisar essas e outras questões importantes.

6.2.1 Endereçamento

Quando um processo de aplicação (por exemplo, um usuário) deseja estabelecer uma conexão com um processo de aplicação remoto, é necessário especificar a aplicação com a qual ele irá se conectar. (O transporte não orientado a conexões tem o mesmo problema: a quem cada mensagem deve ser enviada?) O método normalmente utilizado é definir os endereços de transporte que os processos podem ouvir para receber solicitações de conexão. Na Internet, essas extremidades são chamadas **portas**. Vamos utilizar o termo genérico *ponto de acesso de serviço de transporte*, ou **TSAP** (**Transport Service Access Point**). Os pontos extremos análogos na camada de rede (ou seja, os endereços da camada de rede) são chamados, então, *pontos de acesso de serviço de rede*, ou **NSAPs** (**Network Service Access Points**). Os endereços IP são exemplos de NSAPs.

A Figura 6.5 ilustra o relacionamento entre os NSAPs, os TSAPs e uma conexão de transporte. Os processos de aplicações, tanto clientes quanto servidores, podem se associar a um TSAP para estabelecer uma conexão com um TSAP remoto. Essas conexões funcionam através dos NSAPs de cada host, como mostra a figura. O propósito de ter TSAPs é o fato de, em algumas redes, cada computador ter um único NSAP, e assim necessitar de algum meio para distinguir entre vários pontos extremos de transporte que compartilham esse NSAP.

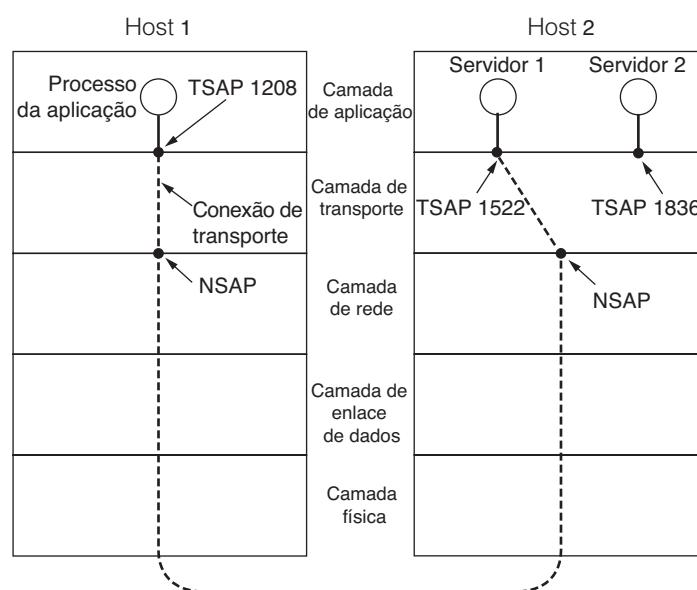



Figura 6.5 | TSAPs, NSAPs e conexões de transporte.

Aqui está um possível cenário para uma conexão de transporte:

1. Um processo servidor de correio se associa ao TSAP 1522 no host 2 para aguardar a chegada de uma chamada. O modo como um processo se associa a um TSAP está fora do escopo do modelo de rede e depende exclusivamente do sistema operacional local. Por exemplo, poderia ser utilizada uma chamada como a nossa primitiva LISTEN.
2. Um processo de aplicação no host 1 deseja enviar uma mensagem de correio e, portanto, transmite uma solicitação CONNECT especificando o TSAP 1208 no host 1 como origem e o TSAP 1522 no host 2 como destino. Em última análise, essa ação resulta no estabelecimento de uma conexão de transporte entre o processo de aplicação e o servidor.
3. O processo de aplicação envia então a mensagem de correio.
4. O servidor de correio responde para dizer que entregará a mensagem.
5. A conexão de transporte é então encerrada (liberada).

Observe que pode haver outros servidores no host 2, associados a outros TSAPs e que estejam esperando pela chegada de conexões de entrada sobre o mesmo NSAP.

O quadro mostrado anteriormente é bom, exceto pelo fato de termos omitido um pequeno problema: como o processo de usuário do host 1 sabe que o servidor de correio está associado ao TSAP 1522? Uma possibilidade é o servidor de correio estar associado ao TSAP 1522 há anos e, gradualmente, todos os usuários da rede terem descoberto isso. Nesse modelo, os serviços têm endereços TSAP estáveis, que estão listados em arquivos guardados em locais conhecidos, como o arquivo */etc/services* dos sistemas UNIX, que lista os servidores que estão associados de modo permanente a cada uma das portas, incluindo o fato de que o servidor de correio se encontra na porta TCP 25.

Embora os endereços TSAP estáveis possam funcionar para um pequeno número de serviços que nunca mudam (por exemplo, o servidor da Web), em geral, os processos do usuário desejam se comunicar com outros processos do usuário que não possuem endereços TSAP conhecidos com antecedência, ou que existem por um curto período de tempo.

Para cuidar dessa situação, um esquema alternativo pode ser utilizado. Nele, existe um processo especial, chamado **portmapper**. Para encontrar o endereço TSAP correspondente a determinado nome de serviço, como 'BitTorrent', um usuário estabelece uma conexão com o portmapper (que escuta em um TSAP conhecido). O usuário envia então uma mensagem especificando o nome do serviço e o portmapper retorna o endereço TSAP. Depois, o usuário encerra a conexão com o portmapper e estabelece uma nova conexão com o serviço desejado.

Nesse modelo, quando um novo serviço é criado, ele precisa se registrar com o portmapper, dando seu nome de serviço (normalmente, uma string ASCII) e seu TSAP. O portmapper registra essa informação em seu banco de dados interno de modo que, quando chegarem consultas mais tarde, ele saberá as respostas.

A função do portmapper é semelhante à de uma telefonista de auxílio à lista no sistema telefônico — ele oferece um mapeamento entre nomes e números. Assim como no sistema telefônico, é essencial que o endereço do TSAP bem conhecido usado pelo portmapper seja realmente bem conhecido. Se você não souber o número da telefonista de informações, não poderá ligar para descobrir o número desejado. Se você acha que o número a ser usado para obter informações é óbvio, tente fazer isso em algum outro país.

Muitos dos processos servidores que podem existir em uma máquina serão usados apenas raramente. É um desperdício ter cada um deles ativo e escutando em um endereço TSAP estável o dia inteiro. Um esquema alternativo aparece na Figura 6.6 em uma forma simplificada. Ele é conhecido como **protocolo de conexão inicial**. Em vez de ter todos os servidores associados a um TSAP conhecido, cada máquina que desejar oferecer serviços a usuários remotos terá um **servidor de processos** especial, que funcionará como um proxy para servidores menos utilizados. Esse servidor é chamado *inetd* em sistemas UNIX. Ele atende a uma série de portas ao mesmo tempo, aguardando uma solicitação de conexão. Os usuários potenciais de um serviço devem começar com uma solicitação CONNECT, especificando o endereço TSAP do serviço que desejam. Se nenhum servidor os estiver aguardando, eles estabelecem uma conexão com o servidor de processos, como mostra a Figura 6.6(a).

Depois de receber a solicitação de entrada, o servidor de processos gera a conexão para o servidor solicitado, permitindo que ele herde a conexão já existente com o usuário. Em seguida, o novo servidor executa a tarefa solicitada, enquanto o servidor de processos volta a aguardar novas solicitações, como mostra a Figura 6.6(b). Esse método só se aplica quando os servidores podem ser criados por demanda.

6.2.2 Estabelecimento de conexões

Estabelecer uma conexão parece uma tarefa fácil, mas, na verdade, trata-se de um procedimento complicado. À primeira vista, pode parecer que basta uma entidade de transporte enviar um segmento CONNECTION REQUEST ao destino e aguardar uma resposta CONNECTION ACCEPTED. O problema é que a rede pode perder, atrasar, corromper e duplicar pacotes. Esse comportamento causa sérias complicações.

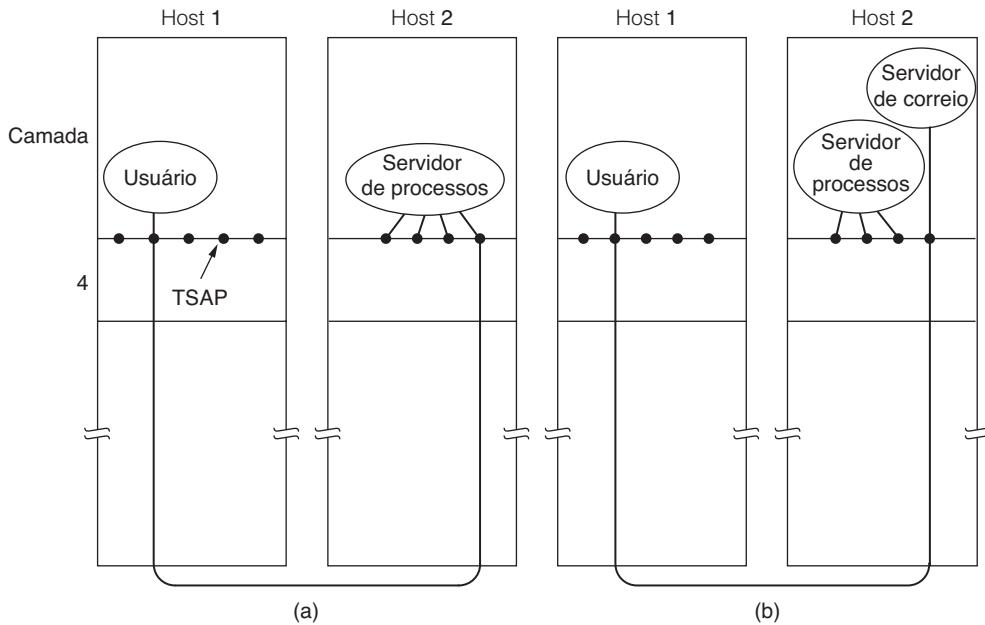



Figura 6.6 | Como um processo do usuário no host 1 estabelece uma conexão com o servidor de correio no host 2 por meio de um servidor de processos.

Imagine uma rede tão congestionada que as confirmações quase nunca chegam a tempo, cada pacote sofre um timeout e é retransmitido duas ou três vezes. Suponha que a rede utilize datagramas internamente e que cada pacote siga uma rota específica. Alguns pacotes podem ficar detidos em um congestionamento de tráfego na rede e demorar muito para chegar, ou seja, eles ficam detidos e só surgem muito depois, quando o transmissor achou que eles tinham sido perdidos.

O pior que pode acontecer é um usuário estabelecer uma conexão com um banco, enviar mensagens dizendo ao banco para transferir uma grande quantia para a conta de uma pessoa não muito confiável. Infelizmente, os pacotes decidem tomar uma rota incomum até o destino e seguem explorando um canto remoto da rede. O transmissor esgota seu tempo-limite e os envia novamente. Dessa vez, os pacotes tomam a rota mais curta e são entregues rapidamente, de modo que o transmissor encerra a conexão.

Infelizmente, o lote inicial de pacotes finalmente chega ao destino na ordem, pedindo ao banco que estabeleça uma nova conexão e transfira o dinheiro (novamente). O banco não tem como saber que se trata de uma duplicata do pedido. Ele supõe que essa é uma segunda transação independente, e transfere o dinheiro mais uma vez.

Esse cenário pode parecer improvável, ou mesmo implausível, mas o importante é que os protocolos precisam ser elaborados para estar corretos em todos os casos. Somente os casos comuns precisam ser implementados de modo eficiente para obter um bom desempenho na rede, mas o protocolo precisa ser capaz de lidar com os casos incomuns sem erros. Se não puder, então teremos montado uma rede razoável, que pode falhar sem aviso quando as condições se tornarem difíceis.

No restante desta seção, vamos estudar o problema das duplicatas atrasadas, dando ênfase especial aos algoritmos para estabelecer conexões confiáveis, de modo a evitar problemas como o que acabamos de descrever. O centro do problema é que as duplicatas atrasadas são consideradas pacotes novos. Não podemos impedir que os pacotes sejam duplicados e atrasados. Mas, se e quando isso acontecer, os pacotes devem ser rejeitados como duplicatas e não processados como pacotes novos.

O problema pode ser combatido de várias maneiras; no entanto, nenhuma delas é muito satisfatória. Uma alternativa possível é usar endereços de transporte descartáveis. Nessa abordagem, toda vez que um endereço de transporte é necessário, um novo endereço é criado. Ao fim da conexão, o endereço é descartado e nunca mais é usado. Os pacotes duplicados atrasados, então, nunca chegam a um processo de transporte e não podem causar danos. Porém, essa alternativa torna mais difícil conectar com um processo em primeiro lugar.

Outra possibilidade é atribuir a cada conexão um identificador (isto é, um número de sequência incrementado a cada conexão estabelecida), escolhido pelo lado que a inicia e colocado em cada segmento, inclusive naquele que solicita a conexão. Após o encerramento, cada entidade de transporte pode atualizar uma tabela que lista as conexões obsoletas como pares (entidade de transporte correspondente, identificador de conexão). Sempre que uma solicitação for recebida, será possível compará-la com a tabela para verificar se ela pertencia a uma conexão já encerrada.

Infelizmente, esse esquema tem uma falha básica: ele exige que cada entidade de transporte mantenha certa quantidade de informações sobre o histórico das conexões por um tempo indeterminado. Esse histórico precisa persistir nas máquinas de origem e de destino. Caso contrário, se uma máquina apresentar defeito e perder sua memória, ela não terá mais como saber quais identificadores de conexões já foram utilizados.

É necessário usar outro método para simplificar o problema. Em vez de permitir que permaneçam na rede eternamente, devemos criar um mecanismo para destruir os pacotes desatualizados que ainda estejam presentes. Com essa restrição, o problema poderá ser contornado.

A duração de um pacote pode ser limitada a um valor máximo conhecido, usando-se uma (ou mais) destas técnicas:

1. Restringir o projeto da rede.
2. Usar um timer de hops em cada pacote.
3. Utilizar um período de tempo em cada pacote.

A primeira técnica inclui qualquer método que evite que um pacote execute um loop, combinado com algum meio de limitar o atraso devido ao congestionamento sobre o caminho mais longo possível (agora conhecido). Isso é difícil, pois as redes interligadas podem variar desde uma única cidade até um nível internacional. O segundo método consiste em ter um timer de hops, iniciado com algum valor apropriado e decrementado toda vez que o pacote é encaminhado. O protocolo de rede simplesmente descarta qualquer pacote cujo timer de hops chega a zero. O terceiro método exige que cada pacote seja associado ao horário em que foi criado, com a concordância dos roteadores em descartar qualquer pacote mais antigo que um tempo previamente estabelecido. Esse último método exige que os clocks dos roteadores estejam sincronizados, o que não é uma tarefa fácil, e na prática um timer de hops é uma aproximação suficiente.

Na prática, é necessário garantir que não apenas o pacote foi destruído, mas também que todas as suas confirmações também foram destruídas; portanto, introduziremos agora o valor *T*, algum múltiplo pequeno da verdadeira duração máxima do pacote. A duração máxima do pacote é uma constante conservadora para uma rede; para a Internet, esse valor é considerado arbitrariamente como 120 segundos. O múltiplo depende de um protocolo e tem o efeito de tornar *T* mais longo. Se esperarmos um tempo de*T* segundos após um pacote ser enviado, poderemos ter certeza de que todos os vestígios do pacote desapareceram, e que nem ele nem suas confirmações surgirão repentinamente para complicar nosso trabalho.

Limitando o tempo de vida dos pacotes, é possível imaginar uma forma infalível de estabelecer conexões com segurança. O método descrito a seguir foi proposto por Tomlinson (1975) e aprimorado mais tarde por Sunshine e Dalal (1978). Na prática, muitas de suas variações são amplamente utilizadas, inclusive no TCP.

O núcleo do método é fazer com que a origem rotule os segmentos com números de sequência que não serão reutilizados dentro de *T* segundos. O período, *T*, e a taxa de pacotes por segundo determinam o tamanho dos números de sequência. Desse modo, somente um pacote com determinado número de sequência pode estar pendente em determinado momento. As duplicatas desse pacote ainda poderão ocorrer, e elas devem ser descartadas pelo destino. Contudo, não acontece mais de uma duplicata atrasada de um pacote antigo ter o mesmo número de sequência e ser aceita pelo destino.

Para contornar o problema da perda de memória de uma máquina após um desastre, uma possibilidade é exigir que as entidades de transporte fiquem ociosas por *T* segundos após uma recuperação. O período ocioso permitirá que todos os segmentos antigos morram, de modo que o transmissor possa começar novamente com qualquer número de sequência. Porém, em uma rede interligada complexa, *T* pode ser grande, de modo que essa estratégia não é atraente.

Em vez disso, Tomlinson propôs que cada host fosse equipado com um clock. Os clocks dos diferentes hosts não precisam estar sincronizados. Cada clock assume a forma de um timer binário incrementado em intervalos regulares. Além disso, o número de bits no timer deve ser maior ou igual ao número de bits dos números de sequência. Por último, e mais importante, o clock deve continuar funcionando mesmo que o host saia do ar.

Quando uma conexão é estabelecida, os *k* bits de baixa ordem do clock são usados como o número de sequência inicial. Assim, diferentemente de nossos protocolos do Capítulo 3, cada conexão começa a numerar seus segmentos com um número de sequência inicial diferente. O espaço de sequência deve ser tão extenso que, quando os números de sequência começarem a se repetir, os segmentos antigos com o mesmo número de sequência já deverão ter sido destruídos há muito tempo. O relacionamento linear entre o tempo e o número de sequência é ilustrado na Figura 6.7(a). A região proibida mostra os tempos para os quais os números de sequência do segmento são inválidos, impedindo seu uso. Se qualquer segmento for enviado com um número de sequência nessa região, ele poderia ser atrasado e personificar um pacote diferente com o mesmo número de sequência que será emitido pouco depois. Por exemplo, se o host falhar e reiniciar no instante 70 segundos, ele usará os números de sequência iniciais com base no clock para prosseguir de onde parou; o host não começa com um número de sequência inferior na região proibida.

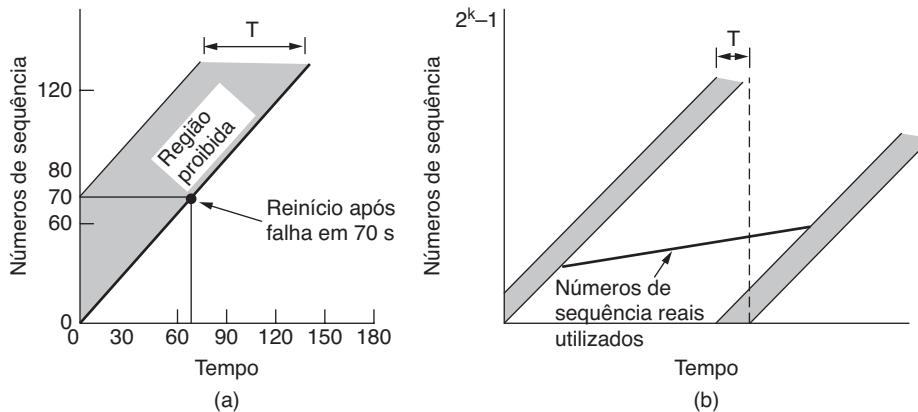



Figura 6.7 | (a) Segmentos não podem entrar na região proibida. (b) O problema da ressincronização.

Uma vez que duas entidades de transporte tenham chegado a um acordo sobre o número de sequência inicial, qualquer protocolo de janela deslizante poderá ser usado para o controle de fluxo de dados. Esse protocolo de janela encontrará e descartará corretamente as duplicatas dos pacotes após já terem sido aceitas. Na verdade, a curva do número de sequência inicial (representada pela linha mais grossa) não é exatamente uma reta, e se parece com uma escada, pois o clock avança em passos discretos. Para simplificar, vamos ignorar esse detalhe.

Para manter os números de sequência de pacote fora da região proibida, precisamos tomar cuidado em dois aspectos. Podemos ter problemas de duas maneiras distintas. Se um host enviar muitos dados muito rapidamente por uma conexão recém-aberta, o número de sequência real versus a curva de tempo pode crescer de forma mais inclinada do que o número de sequência inicial versus a curva de tempo, fazendo com que o número de sequência entre na região proibida. Para impedir que isso aconteça, a taxa de dados máxima em qualquer conexão é de um segmento por pulso de clock. Isso também significa que a entidade de transporte deve esperar até que o pulso de clock ocorra antes de abrir uma nova conexão após um reinício por falha, a menos que o mesmo número seja usado duas vezes. Esses dois pontos são argumentos a favor de um pulso de clock curto (1 μs ou menos). Mas o pulso de clock não pode ser tão rápido quanto o número de sequência. Para uma taxa de clock C em um espaço de número de sequência de tamanho S, devemos ter $S/C>T$, de modo que os números de sequência não possam ser reiniciados muito rapidamente.

Entrar na região proibida por baixo, transmitindo muito rapidamente, não é a única maneira de ter problemas. Pela Figura 6.7(b), vemos que, em qualquer taxa de dados menor que a taxa do clock, a curva dos números de sequência reais usados contra o tempo por fim entrará na região proibida a partir da esquerda, à medida que os números de sequência forem reiniciados. Quanto maior a inclinação dos números de sequência reais, maior o tempo de esse evento ser adiado. Evitar essa situação limita a lentidão com que os números de sequência podem avançar em uma conexão (ou quanto tempo as conexões podem durar).

O método baseado em clock resolve o problema de não poder distinguir os segmentos duplicados e ainda mantidos dos novos segmentos. Porém, existe um problema prático no seu uso para estabelecer conexões. Como normalmente não nos lembramos dos números de sequência entre as conexões no destino, ainda não temos como saber se um segmento CONNECTION REQUEST contendo um número de sequência inicial é uma duplicata de uma conexão recente. Esse problema não existe durante uma conexão, pois o protocolo de janela deslizante se lembra do número de sequência atual.

Para resolver esse problema específico, Tomlinson (1975) criou o **handshake de três vias**. Esse protocolo de estabelecimento de conexão envolve um peer verificando com o outro se a solicitação de conexão está realmente ativa. O procedimento inicial normal quando o host 1 é iniciado aparece na Figura 6.8(a). O host 1 escolhe um número de sequência inicial x e o envia em segmento CONNECTION REQUEST para o host 2. Por sua vez, o host 2 responde com um segmento ACK que confirma x e anuncia seu próprio número de sequência inicial, y. Por fim, o host 1 confirma o número de sequência inicial escolhido pelo host 2 no primeiro segmento de dados que enviar.

Agora, vamos ver como o handshake de três vias funciona diante de duplicatas atrasadas de segmentos de controle. Na Figura 6.8(b), o primeiro segmento é uma duplicata atrasada da primitiva CONNECTION REQUEST de uma antiga conexão. Esse segmento chega ao host 2 sem o conhecimento do host 1. O host 2 reage a esse segmento transmitindo um segmento ACK ao host 1, para verificar se o host 1 deseja realmente estabelecer uma nova conexão. Quando o host 1 rejeita a tentativa de conexão feita pelo host 2, este percebe que foi enganado por uma duplicata atrasada e abandona a conexão. Dessa forma, uma duplicata atrasada não causa danos.

A pior situação ocorre quando tanto uma CONNECTION REQUEST atrasada quanto uma ACK estão pairando na sub-rede. Esse caso é ilustrado na Figura 6.8(c). Como no exemplo anterior, o host 2 recebe uma CONNECTION REQUEST atrasada e responde a ela. É importante entender que o host 2 propôs o uso de *y* como número de sequência inicial para o tráfego do host 2 ao host 1, o que implica não existir nenhum segmento que contenha o número de sequência *y* ou ainda existirem confirmações para *y*. Quando o segundo segmento atrasado chega ao host 2, o fato de *z* ter sido confirmado no lugar de *y* faz com que o host 2 também perceba que se trata de uma duplicata antiga. O importante a observar aqui é que não existe combinação alguma de segmentos antigos que possa fazer o protocolo falhar e configurar uma conexão por acidente quando ela não for solicitada.

O TCP usa esse handshake triplo para estabelecer conexões. Dentro de uma conexão, um período de tempo é usado para estender o número de sequência de 32 bits de modo que não reinicie dentro da duração máxima do pacote, até mesmo para conexões de gigabits por segundo. Esse mecanismo é um reparo para o TCP, que foi necessário para ser usado em enlaces cada vez mais rápidos. Ele é descrito na RFC 1323 e é chamado **PAWS** (**Protection Against Wrapped Sequence numbers**). Entre as conexões, para os números de sequência iniciais e antes de o PAWS ter aparecido, o TCP originalmente usava o esquema baseado em clock, que acabamos de descrever. Porém, descobriu-se que havia uma vulnerabilidade de segurança. O clock tornava fácil para um invasor prever o próximo número de sequência inicial e enviar pacotes que enganavam o handshake triplo e estabeleciam uma conexão falsa. Para fechar essa lacuna, os números de sequência inicial pseudoaleatórios são usados para as conexões na prática. Porém, continua sendo importante que os números de sequência iniciais não se repitam por um intervalo, embora pareçam ser aleatórios a um observador. Caso contrário, as duplicatas atrasadas podem causar problemas.

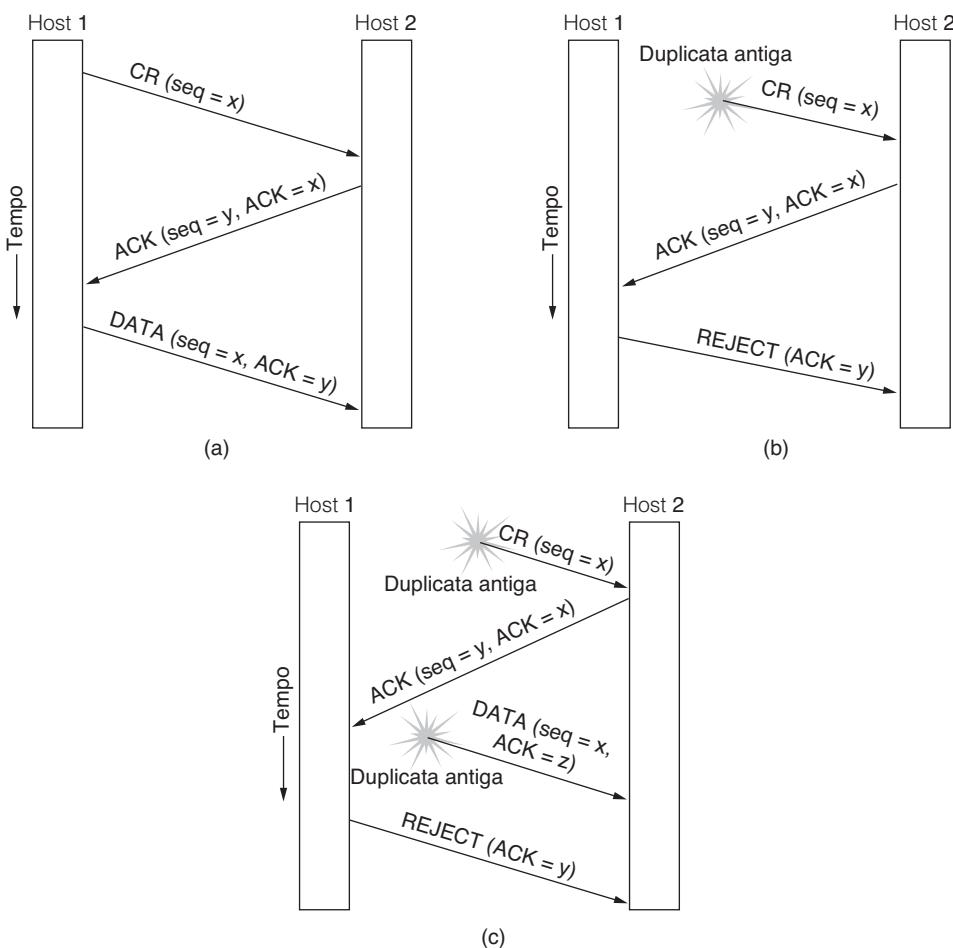



Figura 6.8 | Três cenários de protocolos para o estabelecimento de uma conexão com o uso de um handshake de três vias. CR denota CONNECTION REQUEST. (a) Operação normal. (b) Duplicata antiga de CONNECTION REQUEST que surge repentinamente. (c) CONNECTION REQUEST e ACK duplicadas.

6.2.3 Encerramento de conexões

Encerrar uma conexão é mais fácil do que estabelecê-la. No entanto, nesse procedimento há mais armadilhas do que se poderia esperar. Como já mencionamos, existem dois tipos de encerramento de conexão, o encerramento simétrico e o encerramento assimétrico. O encerramento assimétrico representa o funcionamento do sistema telefônico, ou seja, quando um dos interlocutores desliga, a conexão é interrompida. Em contraste, o encerramento simétrico trata a conexão como duas conexões unidirecionais isoladas e exige que cada uma seja encerrada separadamente.

O encerramento assimétrico é repentino e pode resultar na perda de dados. Considere o cenário da Figura 6.9. Após o estabelecimento da conexão, o host 1 envia um segmento, que chega de forma correta ao host 2. Em seguida, o host 1 envia outro segmento. Infelizmente, o host 2 gera uma primitiva DISCONNECT antes de o segundo segmento chegar, o que resulta no encerramento da conexão e na perda dos dados.

Fica clara a necessidade de utilizar um protocolo de encerramento mais sofisticado para evitar a perda de dados. Uma forma de implementar essa estratégia é usar o encerramento simétrico, no qual cada direção da conexão é liberada de forma independente da outra. Nesse caso, um host pode continuar a receber dados mesmo depois de ter enviado um segmento DISCONNECT.

O encerramento simétrico é indicado quando cada processo tem uma quantidade fixa de dados a enviar e sabe com clareza quando terminou de enviá-los. Em outras situações, não é tão simples determinar que todo o trabalho foi concluído e que a conexão deve ser encerrada. É possível imaginar um protocolo no qual o host 1 diz: ‘Já terminei. Você também já terminou?’. Se o host 2 responder: ‘Eu também já terminei. Adeus.’, a conexão poderá ser encerrada com segurança.

No entanto, nem sempre o protocolo funciona assim. Há um problema famoso que ilustra essa questão, chamado **problema dos dois exércitos**. Imagine que o exército branco está acampado em um vale, conforme ilustra a Figura 6.10. Acampados nas duas colinas ao redor do vale estão dois exércitos azuis. O exército branco tem um contingente maior que cada exército azul, mas juntos os exércitos azuis são maiores que o exército branco. Se um dos exércitos azuis atacar sozinho, ele será derrotado pelo exército branco; porém, se atacarem o exército branco simultaneamente, os dois exércitos azuis serão vitoriosos.

Figura 6.9 | Desconexão repentina com perda de dados.

Os exércitos azuis desejam sincronizar seus ataques. Entretanto, eles só podem se comunicar através de mensageiros que caminham pelo vale, onde podem ser capturados e a mensagem perdida (ou seja, eles têm de usar um canal de comunicação não confiável). A questão é: existe algum protocolo que permita aos exércitos azuis vencer?

Suponhamos que o comandante do exército azul número 1 envie uma mensagem dizendo: ‘Proponho que nosso ataque seja ao amanhecer do dia 29 de março. Que tal?’. Suponhamos também que a mensagem chegue a seu destino e que o comandante do exército azul número 2 concorde, e que sua mensagem chegue com segurança ao exército azul número 1. O ataque acontecerá? Provavelmente não, pois o comandante número 2 não tem como saber se o comandante número 1 recebeu sua resposta. Caso não tenha recebido a resposta, ele não vai atacar; portanto, seria inútil o exército azul número 2 iniciar a batalha sozinho.

Agora, vamos melhorar o protocolo tornando-o um handshake de três vias. O responsável pela proposta original deve confirmar a resposta fornecida. Supondo que nenhuma mensagem se perca, o exército azul número 2 receberá a confirmação. No entanto, agora é a vez de o comandante do exército número 1 hesitar. Afinal, ele não sabe se sua confirmação chegou ao destino e assim o exército azul número 2 não vai atacar. Poderíamos criar aqui o handshake de quatro vias, mas isso também não ajudaria.

Na verdade, podemos provar que não existe nenhum protocolo que funcione. No entanto, suponhamos que esse protocolo existisse. Nesse caso, deveríamos verificar se a última mensagem do protocolo é essencial ou não. Se a resposta for negativa, devemos removê-la (assim como todas as outras mensagens que não forem essenciais) até ficarmos com um protocolo no qual toda mensagem seja essencial. O que acontecerá se a mensagem final não chegar a seu destino? Acabamos de afirmar que ela é essencial; portanto, se essa mensagem se perder, o ataque não ocorrerá. Como o transmissor não tem certeza de que a mensagem final foi recebida, ele não arrisca um ataque. Pior ainda, o outro exército azul sabe disso e também não ataca.

Para compreender a relevância do problema dos dois exércitos no encerramento de conexões, basta substituir ‘atacar’ por ‘desconectar’. Se nenhum dos lados estiver preparado para encerrar a conexão até estar convencido de que o outro lado também está pronto, o encerramento jamais acontecerá.

Na prática, podemos evitar esse dilema deixando de lado a necessidade de um acordo e empurrando o problema para o usuário do transporte, deixando que cada lado

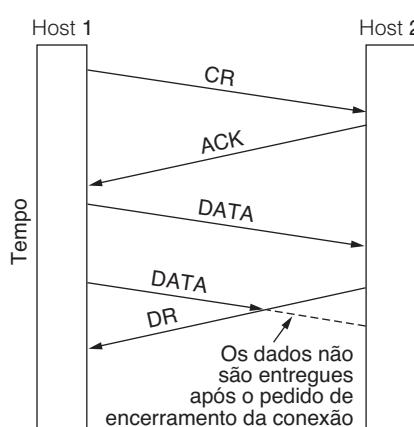
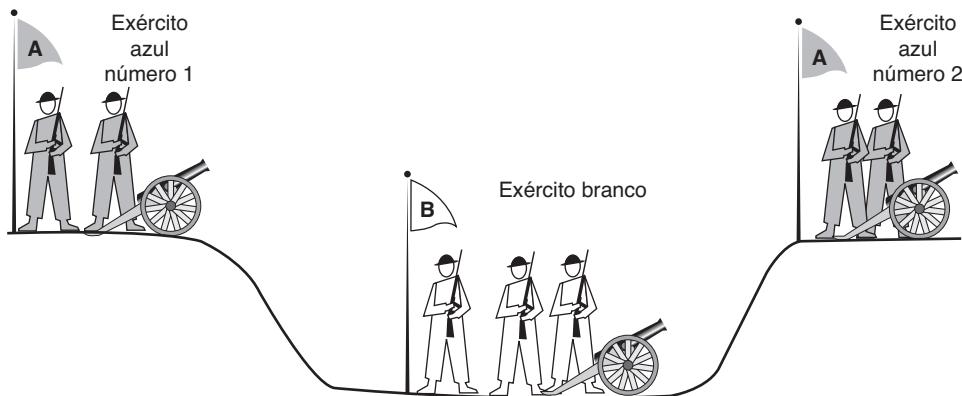



Figura 6.10 | O problema dos dois exércitos.

decida independentemente de quando isso será feito. Esse é um problema mais fácil de ser resolvido. A Figura 6.11 ilustra quatro cenários de encerramento utilizando o handshake de três vias. Ainda que não seja infalível, em geral esse protocolo é o mais adequado.

Na Figura 6.11(a), vemos uma situação normal em que um dos usuários envia um segmento DR (DISCONNECTION REQUEST) para dar início ao encerramento da conexão. Quando o segmento chega, o receptor também retorna um segmento DR e dispara um timer para o caso de a DR se perder. Quando essa DR chega, o transmissor original envia um segmento ACK e encerra a conexão. Finalmente, quando um segmento ACK chega, o receptor também encerra a conexão. Encerrar uma conexão significa que a entidade de transporte remove as informações sobre essa conexão de sua tabela de conexões atualmente abertas e envia algum tipo de sinal ao proprietário da conexão (o usuário de transporte). Essa ação é diferente de um usuário de transporte emitir uma primitiva DISCONNECT.

Se o último segmento ACK for perdido, como ilustra a Figura 6.11(b), a situação poderá ser salva pelo timer. Quando o timer expirar, a conexão será encerrada de qualquer forma.

Agora, vamos considerar o caso em que a segunda DR se perde. O usuário que der início à desconexão não receberá a resposta esperada, sofrerá um timeout e terá de começar tudo outra vez. Na Figura 6.11(c), podemos ver como isso funciona, supondo que da segunda vez nenhum segmento se perdeu e que todos os segmentos foram entregues de forma correta e em tempo.

Nosso último cenário, a Figura 6.11(d), é igual ao da Figura 6.11(c), exceto pelo fato de assumirmos agora que todas as tentativas repetidas de retransmitir a DR também falharam devido a segmentos perdidos. Após *N* tentativas, o transmissor desiste e encerra a conexão. Enquanto isso, o receptor sofre um timeout e também para de funcionar.

Apesar de em geral ser suficiente, na teoria esse protocolo pode falhar, se a DR inicial e todas as *N* retransmissões se perderem. O transmissor desistirá e encerrará a conexão, enquanto o outro lado não saberá nada sobre todas as tentativas de desconexão e permanecerá ativo. Isso resulta em uma conexão semiaberta.

Poderíamos evitar esse problema não permitindo ao transmissor desistir após *N* tentativas e forçando-o a continuar tentando até obter resposta. Entretanto, se o outro lado tiver permissão para entrar em timeout, o transmissor continuará tentando para sempre, pois nenhuma resposta será recebida. Se não permitirmos que o lado receptor entre em timeout, o protocolo se comportará como na Figura 6.11(d).

Uma forma de extinguir conexões semiabertas é criar uma regra informando que, se nenhum segmento chegar durante determinado número de segundos, a conexão será encerrada automaticamente. Dessa forma, se um lado encerrar a conexão, o outro perceberá a falta de atividade e também encerrará a conexão. Essa regra também cuida do caso em que a conexão é interrompida (pois a rede não pode mais entregar pacotes entre os hosts) sem que alguma extremidade desconecte primeiro. É obvio que, se essa regra for utilizada, será necessário que cada entidade de transporte tenha um timer que será interrompido e depois reiniciado sempre que um segmento for enviado. Se esse timer expirar, um segmento fictício será transmitido para evitar que o outro lado se desconecte. Por outro lado, se a regra de encerramento automático for usada e muitos segmentos fictícios seguidos se perderem em uma conexão inativa, a transmissão será encerrada automaticamente, um lado de cada vez.

Não trataremos mais desse assunto, mas já deve estar claro que encerrar uma conexão sem perda de dados não é tão simples quanto parecia à primeira vista. A lição que aprendemos aqui é que o usuário do transporte precisa estar envolvido na decisão de quando desconectar — o problema não pode ser resolvido de forma limpa pelas próprias entidades de transporte. Para ver a importância da aplicação, considere que, embora o TCP normalmente realize um

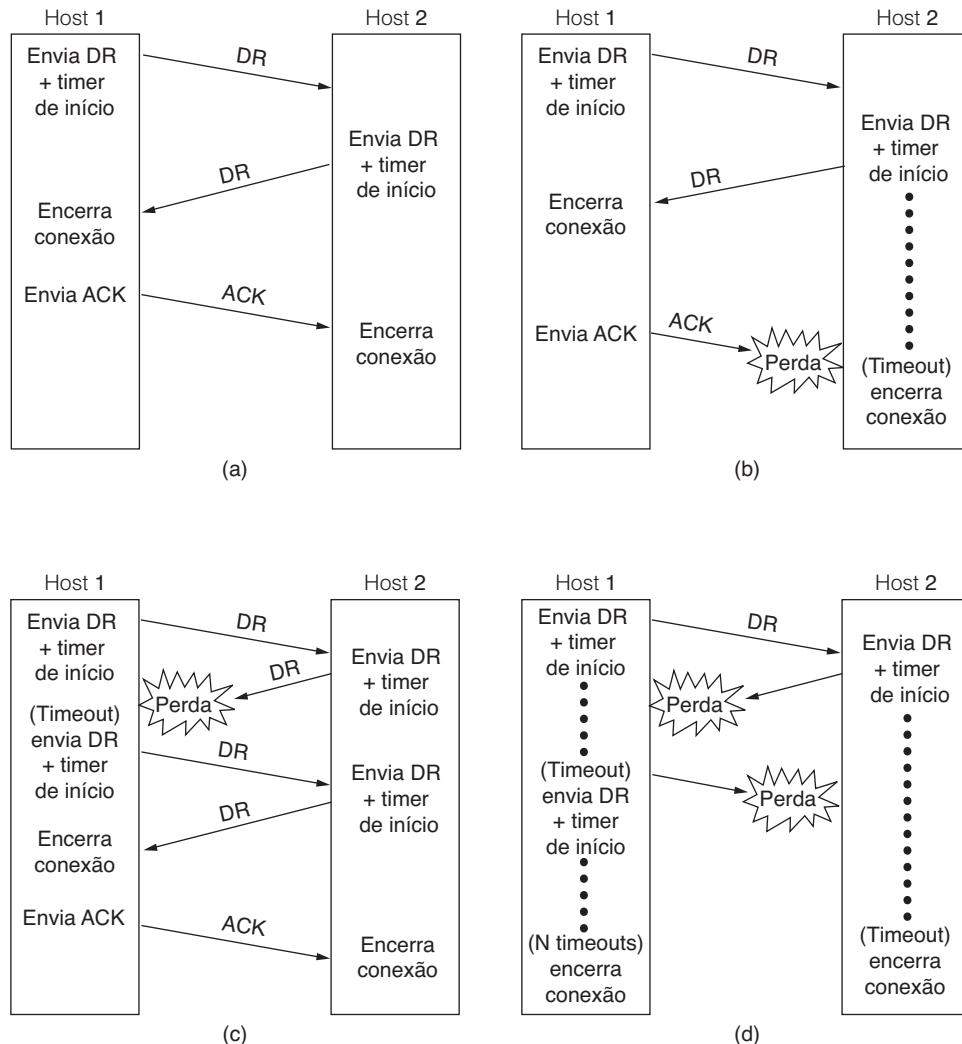



Figura 6.11 | Quatro situações de protocolos para encerramento de uma conexão. (a) Caso normal de handshake de três vias. (b) ACK final perdida. (c) Resposta perdida. (d) Resposta perdida e DRs subsequentes perdidas.

encerramento simétrico (com cada lado encerrando independentemente sua metade da conexão com uma flag FIN quando tiver enviado seus dados), muitos servidores Web enviam ao cliente um pacote com uma flag RST que causa um encerramento brusco da conexão, mais semelhante a um fechamento assimétrico. Isso só funciona porque o servidor Web conhece o padrão da troca de dados. Primeiro, ele recebe uma solicitação do cliente, que são todos os dados que o cliente enviará, e depois envia uma resposta ao cliente. Quando o servidor Web termina com sua resposta, todos os dados foram enviados em ambas as direções. O servidor pode enviar um aviso ao cliente e encerrar a conexão repentinamente. Se o cliente receber esse aviso, ele encerrará o estado de sua conexão nesse momento. Se o cliente não receber o aviso, ele por fim notará que o servidor não está mais falando com ele e encerrará a conexão. De qualquer forma, os dados terão sido transferidos com sucesso.

6.2.4 Controle de erro e controle de fluxo

Tendo examinado o estabelecimento e o encerramento da conexão com algum detalhe, agora veremos como as conexões são gerenciadas enquanto estão em uso. As principais questões são controle de erro e controle de fluxo. Controle de erro é garantir que os dados sejam entregues com o nível de confiabilidade desejado, normalmente que todos os dados sejam entregues sem nenhum erro. Controle de fluxo é impedir que um transmissor rápido sobrecarregue um receptor lento.

Essas duas questões já apareceram antes, quando estudamos a camada de enlace de dados. As soluções usadas na camada de transporte têm os mesmos mecanismos que estudamos no Capítulo 3. Como uma recapitulação muito rápida:

1. Um quadro transporta um código de detecção de erro (por exemplo, um CRC ou checksum) que é usado para verificar se a informação foi recebida corretamente.

2. Um quadro transporta um número de sequência para se identificar e ser retransmitido pelo transmissor até que receba uma confirmação de recebimento bem-sucedido do receptor. Isso é chamado **ARQ** (**Automatic Repeat reQuest**).
3. Existe um número máximo de quadros que o transmissor permitirá que estejam pendentes ao mesmo tempo, interrompendo se o receptor não estiver confirmando quadros com rapidez suficiente. Se esse máximo for um pacote, o protocolo é chamado **stop-and-wait**. Janelas maiores permitem a canalização e melhoram o desempenho sobre enlaces longos e rápidos.
4. O protocolo de **janela deslizante** combina esses recursos e também é usado para dar suporte à transferência de dados bidirecional.

Visto que esses mecanismos são usados em quadros na camada de enlace, é natural questionar por que eles também seriam usados em segmentos na camada de transporte. Porém, na prática, há pouca duplicação entre as camadas de enlace e transporte. Embora os mesmos mecanismos sejam utilizados, existem diferenças em função e em grau.

Para uma diferença em função, considere a detecção de erro. O checksum da camada de enlace protege um quadro enquanto ele atravessa um único enlace. O checksum da camada de transporte protege um segmento enquanto ele atravessa um caminho de rede inteiro. Essa é uma verificação fim a fim, que não é o mesmo que ter uma verificação em cada enlace. Saltzer et al. (1984) descrevem uma situação em que os pacotes foram corrompidos dentro de um roteador. Os checksums da camada de enlace protegeram os pacotes enquanto eles atravessaram um enlace, e não enquanto eles estavam dentro do roteador. Assim, os pacotes foram entregues incorretamente, embora estivessem corretos segundo as verificações em cada enlace.

Esse e outros exemplos levaram Saltzer et al. a articular o **argumento fim a fim**. De acordo com esse argumento, a verificação da camada de transporte que é executada fim a fim é essencial para a exatidão, e as verificações da camada de enlace não são essenciais, mas apesar disso são valiosas para melhorar o desempenho (pois sem elas um pacote corrompido pode ser enviado pelo caminho inteiro desnecessariamente).

Como uma diferença em grau, considere as retransmissões e o protocolo de janela deslizante. A maioria dos enlaces sem fio, diferentemente dos enlaces de satélite, pode ter apenas um único quadro pendente a partir do transmissor de uma só vez. Ou seja, o produto largura de banda-atraso para o enlace é tão pequeno que nem sequer um quadro inteiro pode ser armazenado dentro do enlace. Nesse caso, um tamanho de janela pequeno é suficiente para o bom desempenho. Por exemplo, o 802.11 usa um protocolo stop-and-wait, transmitindo ou retransmitindo cada quadro e esperando que seja confirmado antes de prosseguir para o quadro seguinte. Ter um tamanho de janela maior que um quadro aumentaria a complexidade sem melhorar o desempenho. Para enlaces com fios e de fibra óptica, como a Ethernet (comutada) ou backbones de ISP, a taxa de erros é baixa o suficiente para que as retransmissões da camada de enlace possam ser omitidas, pois as retransmissões fim a fim repararão a perda de quadros residual.

Por outro lado, muitas conexões TCP têm um produto largura de banda-atraso que é muito maior que um único segmento. Considere uma conexão enviando dados pelos Estados Unidos a 1 Mbps com um tempo de ciclo de 100 ms. Até mesmo para essa conexão lenta, 200 Kbits de dados serão armazenados no receptor no tempo necessário para enviar um segmento e receber uma confirmação. Para essas situações, uma grande janela deslizante deve ser usada e o stop-and-wait prejudicará o desempenho. Em nosso exemplo, isso limitaria o desempenho a um segmento a cada 200 ms, ou 5 segmentos/s, não importa quão veloz a rede realmente seja.

Dado que os protocolos de transporte geralmente usam janelas deslizantes maiores, veremos a questão do buffering de dados com mais cuidado. Como um host pode ter muitas conexões, cada uma tratada separadamente, ele pode precisar de uma quantidade substancial de buffering para as janelas deslizantes. Os buffers são necessários no transmissor e no receptor. Certamente, eles são necessários no transmissor para manter todos os segmentos transmitidos mas ainda não confirmados. Eles são necessários lá porque esses segmentos podem ser perdidos e precisam ser retransmitidos.

Porém, como o transmissor está mantendo buffers, o receptor pode ou não dedicar buffers específicos a conexões específicas, como desejar. O receptor pode, por exemplo, manter um único pool de buffers compartilhado por todas as conexões. Quando um segmento chega, é feita uma tentativa de adquirir dinamicamente um novo buffer. Se houver um disponível, o segmento é aceito; caso contrário, ele é descartado. Como o transmissor está preparado para retransmitir segmentos perdidos pela rede, nenhum prejuízo permanente será causado se o receptor descartar segmentos, embora alguns recursos sejam desperdiçados. O transmissor simplesmente continua tentando até receber uma confirmação.

A melhor alternativa entre buffering na origem e buffering no destino depende do tipo de tráfego transportado pela conexão. Para um tráfego em rajada, com baixa largura de banda, como aquele produzido por um terminal interativo, é razoável não dedicar nenhum buffer, mas adquiri-lo dinamicamente nas duas extremidades, confiando no buffering no transmissor se os segmentos ocasionalmente tiverem de ser descartados. Por outro lado, para a transferência de arquivos e outro tráfego com alta largura de banda, é melhor que o receptor não dedique uma janela inteira de buffers, mas permita que os dados fluam em velocidade máxima. Essa é a estratégia usada pelo TCP.

Ainda resta a questão de como organizar o pool de buffers. Se a maioria dos segmentos for quase do mesmo tamanho, é natural organizar os buffers como um pool de buffers de tamanho idêntico, com um segmento por buffer, como na Figura 6.12(a). Porém, se houver uma grande variação no tamanho do segmento, de solicitações curtas para páginas Web e pacotes grandes em transferências de arquivo peer-to-peer, um pool de buffers de tamanho fixo apresentará problemas. Se o tamanho do buffer for escolhido para ser igual ao maior tamanho de segmento possível, o espaço será desperdiçado sempre que um segmento curto chegar. Se o tamanho do buffer for escolhido para ser menor que o tamanho máximo do segmento, vários buffers serão necessários para segmentos longos, com a complexidade criada nesse caso.

Outra abordagem para o problema do tamanho dos buffers é usar buffers de tamanho variável, como mostra a Figura 6.12(b). A vantagem nesse caso é a melhor utilização de memória, à custa de um gerenciamento de buffers mais complicado. Uma terceira possibilidade é dedicar um grande buffer reservado a cada conexão, como ilustra a Figura 6.12(c). Esse sistema é simples e elegante, e não depende do tamanho dos segmentos, mas só faz um bom uso da memória quando todas as conexões têm uma carga muito pesada.

À medida que as conexões são abertas e fechadas e que o padrão de tráfego se altera, o transmissor e o receptor precisam ajustar dinamicamente suas alocações de buffers. Consequentemente, o protocolo de transporte deve permitir que o host transmissor solicite espaço em buffer na outra extremidade da conexão. Os buffers poderiam ser alocados por conexão ou coletivamente, para todas as conexões em execução entre os dois hosts. Em contrapartida, o receptor, conhecendo a situação de seus buffers (mas sem conhecer o tráfego oferecido), poderia dizer ao transmissor: 'Tenho *X* buffers reservados para você'. Se o número de conexões abertas aumentar, talvez seja necessário reduzir uma alocação de buffer; portanto, o protocolo deve oferecer essa possibilidade.

Um modo razoavelmente genérico de gerenciar a alocação dinâmica de buffers é desvincular o gerenciamento dos buffers das confirmações, ao contrário do que ocorre com os protocolos de janela deslizante do Capítulo 3. Na verdade, o gerenciamento dinâmico de buffers significa usar uma janela de tamanho variável. Inicialmente, o transmissor solicita determinado número de buffers, com base em suas necessidades. Em seguida, de acordo com o número solicitado, o receptor oferece todos os buffers de que dispõe. Sempre que enviar um segmento, o transmissor deverá decrementar sua alocação, parando por completo quando a alocação chegar a zero. Em seguida, o receptor transmite (por piggyback) as confirmações e as alocações de buffers no tráfego reverso. O TCP usa esse esquema, transportando alocações de buffer em um campo do cabeçalho chamado *Tamanho de janela*.

A Figura 6.13 mostra um exemplo de como o gerenciamento dinâmico de janelas poderia funcionar em uma rede de datagramas com números de sequência de 4 bits. Nesse exemplo, os dados fluem em segmentos do host *A* para o host *B* e as confirmações e alocações de buffers fluem em segmentos no sentido contrário. Inicialmente, *A* quer oito buffers, dos quais só recebe quatro. Em seguida, *A* envia três segmentos, sendo que o terceiro é perdido. O segmento 6 confirma a recepção de todos os segmentos até o número de sequência 1, inclusive, permitindo que *A* libere esses buffers, e informa a *A* que ele tem permissão para enviar mais três segmentos, começando pelo segmento se-

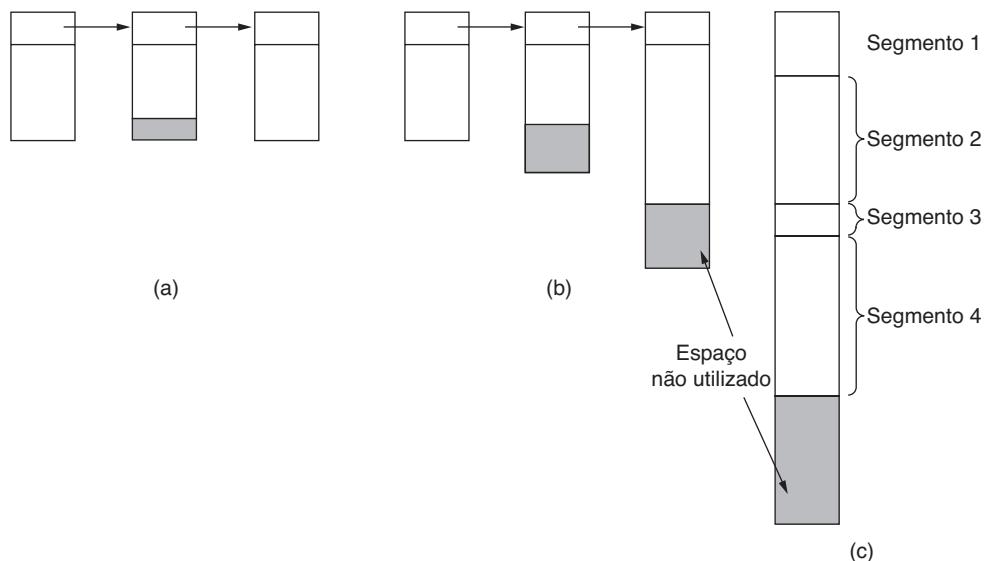



Figura 6.12 | (a) Buffers de tamanho fixo encadeados. (b) Buffers de tamanho variável encadeados. (c) Um grande buffer reservado por conexão.

guinte ao de número 1 (isto é, os segmentos 2, 3 e 4). *A* sabe que já enviou o segmento de número 2 e assim imagina que pode enviar os segmentos 3 e 4, o que acaba por fazer. Nesse ponto, ele é bloqueado e deve aguardar a alocação de mais buffers. Entretanto, pode haver retransmissões (linha 9) induzidas por timeouts durante o bloqueio, pois elas utilizam buffers que já haviam sido alocados. Na linha 10, *B* confirma a recepção de todos os segmentos até 4, inclusive, mas se recusa a permitir que *A* continue. Essa situação é impossível com os protocolos de janela fixa do Capítulo 3. O próximo segmento de *B* para *A* aloca outro buffer e permite que A continue. Isso acontecerá quando *B* tem espaço em buffer, provavelmente porque o usuário do transporte aceitou mais segmentos de dados.

Podem surgir problemas potenciais com esquemas de alocação de buffers desse tipo em redes de datagramas, caso ocorra a perda de segmentos de controle — e eles certamente surgem. Observe a linha 16. *B* já alocou mais buffers para *A*, mas o segmento da alocação foi perdido. Como os segmentos de controle não respeitam uma sequência nem sofrem timeout, *A* está em um impasse. Para evitar que isso aconteça, cada host deve enviar periodicamente segmentos de controle informando o status dos buffers e das confirmações em cada conexão. Dessa forma, o impasse será resolvido mais cedo ou mais tarde.

Até agora, partimos da suposição tácita de que o único limite imposto sobre a taxa de dados do transmissor é o espaço em buffer disponível no receptor. Isso normalmente não acontece. A memória já foi muito cara, mas os preços caíram bastante. Os hosts podem ser equipados com memória suficiente, de modo que a falta de buffers raramente ou nunca será um problema, até mesmo para conexões remotas. É claro que isso depende do tamanho do buffer sendo definido como grande o suficiente, o que nem sempre aconteceu no caso do TCP (Zhang et al., 2002).

Quando o espaço em buffer deixar de limitar o fluxo máximo, surgirá outro gargalo: a capacidade de transporte da rede. Se os roteadores adjacentes puderem trocar dados em uma taxa de no máximo x pacotes/s, e se houver k caminhos disjuntos entre um par de hosts, esses hosts não poderão trocar mais de kx segmentos/s, independentemente do espaço em buffer disponível em cada extremidade da conexão. Se o transmissor forçar muito a transmissão (ou seja, enviar mais de kx segmentos/segundo), a rede ficará congestionada, pois não será capaz de entregar os segmentos na mesma taxa em que eles chegam.

É necessário um mecanismo que limite as transmissões com base na capacidade de transporte da rede, em vez da capacidade dos buffers do receptor. Belsnes (1975) propôs o uso de um esquema de controle de fluxo com uma janela deslizante, no qual o transmissor ajusta dinamicamente o tamanho da janela à capacidade de transporte da rede. Isso significa que o tamanho da janela deslizante pode implementar controle de fluxo e contro-

	A	Mensagem	B	Comentários
1	→	<request 8 buffers>	→	A deseja 8 buffers
2	←	<ack = 15, buf = 4>	←	B concede mensagens 0-3 apenas
3	→	<seq = 0, data = m0>	→	A tem 3 buffers restantes agora
4	→	<seq = 1, data = m1>	→	A tem 2 buffers restantes agora
5	→	<seq = 2, data = m2>	•••	Mens. perdida, mas A acha que tem 1 restante
6	←	<ack = 1, buf = 3>	←	B confirma 0 e 1, permite 2-4
7	→	<seq = 3, data = m3>	→	A tem 1 buffer restante
8	→	<seq = 4, data = m4>	→	A tem 0 buffers restantes e deve parar
9	→	<seq = 2, data = m2>	→	A atinge o timeout e retransmite
10	←	<ack = 4, buf = 0>	←	Tudo confirmado, mas A ainda bloqueado
11	←	<ack = 4, buf = 1>	←	A pode agora enviar 5
12	←	<ack = 4, buf = 2>	←	B encontrou novo buffer em algum lugar
13	→	<seq = 5, data = m5>	→	A tem 1 buffer restante
14	→	<seq = 6, data = m6>	→	A agora está bloqueado novamente
15	←	<ack = 6, data = m6>	←	A ainda está bloqueado
16	•••	<ack = 6, buf = 4>	←	Impasse em potencial

Figura 6.13 | Alocação dinâmica de buffers. As setas mostram o sentido da transmissão. As reticências (...) indicam a perda de um segmento.

le de congestionamento. Se a rede puder administrar *c* segmentos/s e o tempo de ciclo (incluindo transmissão, propagação, enfileiramento, processamento no receptor e retorno da confirmação) for *r*, a janela do transmissor deverá ser *cr*. Com uma janela desse tamanho, o transmissor normalmente opera com toda a capacidade do pipeline. Qualquer pequena diminuição no desempenho da rede bloqueará o fluxo. Como a capacidade de rede disponível para qualquer fluxo varia com o tempo, o tamanho da janela deverá ser ajustado com frequência, para acompanhar as mudanças na capacidade de transporte. Como veremos mais adiante, o TCP usa um esquema semelhante.

6.2.5 Multiplexação

A multiplexação, ou compartilhamento de várias conversações em conexões, circuitos virtuais e enlaces físicos, tem um papel importante em diversas camadas da arquitetura de rede. Na camada de transporte, a necessidade da multiplexação pode surgir de diversas formas. Por exemplo, se houver apenas um endereço de rede disponível em um host, todas as conexões de transporte nessa máquina terão de utilizá-lo. Ao chegar um segmento, é necessário encontrar algum meio de descobrir a qual processo ele deve ser entregue. Essa situação, denominada **multiplexação**, é ilustrada na Figura 6.14(a). Nessa figura, quatro conexões de transporte distintas utilizam a mesma conexão de rede (por exemplo, um endereço IP) para o host remoto.

A multiplexação também pode ser útil na camada de transporte por outra razão. Por exemplo, suponha que um host tenha vários caminhos de rede que ele possa usar. Se um usuário necessitar de maior largura de banda ou mais confiabilidade do que um dos caminhos de rede pode fornecer, uma saída será uma conexão que distribua o tráfego entre vários caminhos de rede em rodízio, como indica a Figura 6.14(b). Esse modo de operação é chamado **multiplexação inversa**. Com *k* conexões de rede abertas, a largura de banda efetiva é aumentada *k* vezes. Um exemplo comum de multiplexação inversa é o **SCTP** (**Stream Control Transmission Protocol**), que pode trabalhar com uma conexão usando várias interfaces de rede. Ao contrário, o TCP utiliza uma única extremidade de rede. A multiplexação inversa também é encontrada na camada de enlace, quando vários enlaces de baixa velocidade são usados em paralelo como um enlace de alta velocidade.

6.2.6 Recuperação de falhas

Se os hosts e roteadores estiverem sujeitos a interrupções em seu funcionamento ou se as conexões tiverem longa vida (por exemplo, download grande de software ou multimídia), a recuperação dessas panes se tornará uma questão importante. Se a entidade de transporte estiver inteiramente contida nos hosts, a recuperação de falhas na rede ou no roteador será simples. As entidades de transporte esperam segmentos perdidos o tempo todo e sabem como lidar com eles usando retransmissões.

Um problema mais complexo é como recuperar o funcionamento depois de uma pane no host. Em particular, talvez seja preferível que os clientes possam continuar funcionando quando os servidores falharem e forem rapidamente reiniciados em seguida. Para ilustrar a dificuldade, vamos supor que um host, o cliente, esteja enviando um arquivo muito grande a outro host, o servidor de arquivos, utilizando um protocolo stop-and-wait simples. A camada de transporte do servidor simplesmente passa os segmentos que chegam para o usuário de transporte, um a um. No meio da transmissão, o servidor sai do ar. Quando ele volta a funcionar, suas tabelas são reiniciadas, e ele não sabe mais exatamente onde estava.

Na tentativa de recuperar seu estado anterior, o servidor pode transmitir um segmento por broadcast a todos os outros hosts, comunicando seu problema e solicitando que seus clientes o informem sobre o status de todas as

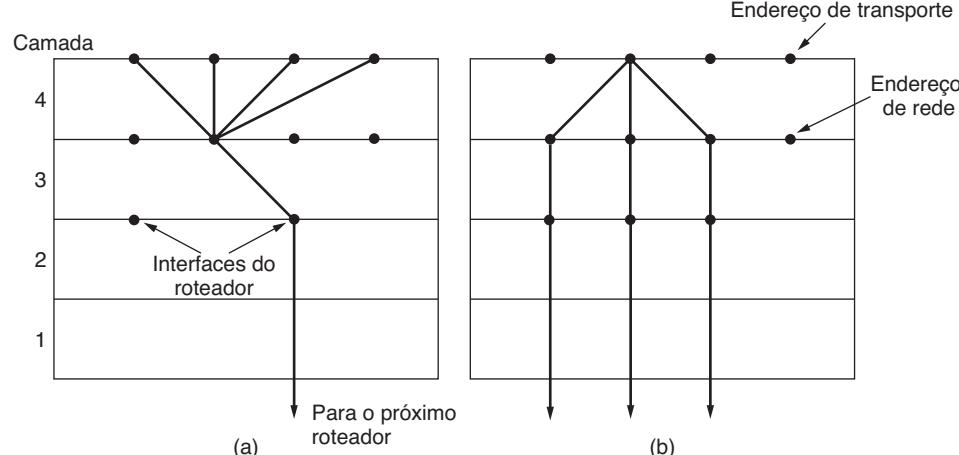



Figura 6.14 | (a) Multiplexação. (b) Multiplexação inversa.

conexões abertas. Cada cliente pode estar em uma das seguintes situações: um segmento pendente, *S1*, ou nenhum segmento pendente, *S0*. Com base apenas nessas informações de estado, o cliente tem de decidir se deve ou não retransmitir o segmento mais recente.

À primeira vista, parece óbvio que o cliente só deve retransmitir se houver um segmento não confirmado pendente (isto é, se ele se encontrar no estado *S1*) durante a queda do servidor. Contudo, uma análise mais detalhada revela as dificuldades dessa abordagem simplista. Por exemplo, considere a situação na qual a entidade de transporte do servidor primeiro envia uma confirmação e depois, quando a confirmação tiver sido enviada, efetua uma gravação no processo de aplicação. Gravar um segmento no fluxo de saída e enviar uma confirmação são dois eventos distintos e indivisíveis que não podem ser realizados simultaneamente. Se ocorrer uma falha após o envio da confirmação, mas antes de a gravação ser feita, o cliente receberá a confirmação e assim ficará no estado *S0* quando chegar o anúncio de que o funcionamento foi recuperado. Consequentemente, o cliente não retransmitirá, porque imagina (incorretamente) que o segmento chegou. Essa decisão do cliente ocasiona a perda de um segmento.

A essa altura, você deve estar pensando: 'É fácil resolver esse problema. Basta reprogramar a entidade de transporte para gravar primeiro o segmento e depois enviar a confirmação'. Tente outra vez. Imagine que a gravação foi feita mas a falha do servidor ocorreu antes de ser possível enviar a confirmação. O cliente estará no estado *S1* e, portanto, retransmitirá, acarretando uma duplicata do segmento não detectada no fluxo de saída para o processo de aplicação do servidor.

Independentemente da forma como o cliente e o servidor são programados, sempre haverá situações em que o protocolo não recuperará o funcionamento de modo apropriado. O servidor poderá ser programado de duas maneiras: para confirmar primeiro ou para gravar primeiro. O cliente poderá ser programado de quatro formas: para sempre retransmitir o último segmento, para nunca retransmitir o último segmento, para retransmitir somente no estado *S0* ou para retransmitir somente no estado *S1*. Isso nos dá oito combinações, mas, como veremos, para cada combinação existe algum conjunto de eventos que faz o protocolo falhar.

Há três eventos possíveis no servidor: enviar uma confirmação (A), gravar no processo de saída (W) e sofrer uma pane (C). Os três eventos podem ocorrer em seis ordens distintas: $AC(W)$, AWC, $C(AW)$, $C(WA)$, WAC e $WC(A)$, onde os parênteses são usados para indicar que nem A nem W podem seguir C (ou seja, não há nenhum evento após uma pane). A Figura 6.15 mostra todas as oito combinações de estratégias do cliente e do servidor, e as sequências de eventos válidas para cada uma delas. Observe que, para cada estratégia, existe alguma sequência de eventos que resulta na falha do protocolo. Por exemplo, se o cliente sempre retransmitir, o evento AWC gerará uma duplicata não detectada, mesmo que os dois outros eventos funcionem perfeitamente.

Tornar o protocolo mais elaborado também não ajuda muito. Ainda que o cliente e o servidor troquem vários segmentos antes de o servidor tentar gravar, de forma que o cliente saiba exatamente o que está para acontecer, o cliente não terá meios para saber se ocorreu uma pane imediatamente antes ou imediatamente após a gravação. A conclusão é inevitável: sob nossas regras básicas de não haver eventos simultâneos — ou seja, eventos separados acontecem um após o outro, e não ao mesmo tempo —, a queda e a recuperação do host não podem ser realizadas de forma transparente para as camadas mais altas.

Em termos mais genéricos, esse resultado pode ser reafirmado como o fato de que a recuperação de uma queda da camada N só pode ser feita pela camada $N + 1$, e mesmo assim somente se a camada mais elevada mantiver um volume suficiente de informações sobre o *status* para reconstruir

Estratégia usada pelo host receptor

Estratégia usada pelo host transmissor	Primeiro ACK, depois gravar			Primeiro gravar, depois ACK		
	AC(W)	AWC	C(AW)	C(WA)	WAC	WC(A)
Sempre retransmitir	OK	DUP	OK	OK	DUP	DUP
Nunca retransmitir	LOST	OK	LOST	LOST	OK	OK
Retransmitir em S0	OK	DUP	LOST	LOST	DUP	OK
Retransmitir em S1	LOST	OK	OK	OK	OK	DUP

OK = Protocolo funciona corretamente
DUP = Protocolo gera uma mensagem duplicada
LOST = Protocolo perde uma mensagem

Figura 6.15 | Diferentes combinações de estratégias do cliente e do servidor.

onde estava antes que o problema ocorresse. Como mencionamos anteriormente, a camada de transporte pode se recuperar de falhas na camada de rede, desde que cada extremidade da conexão tenha uma ideia do ponto em que está.

Esse problema nos leva à questão do que significa de fato a chamada confirmação fim a fim. Em princípio, o protocolo de transporte é fim a fim, pois não é encadeado como as camadas inferiores. Considere agora o caso de um usuário que solicita transações relativas a um banco de dados remoto. Suponha que a entidade de transporte remota esteja programada de modo a passar primeiro os segmentos para a camada imediatamente superior e só então enviar a confirmação. Até mesmo nesse caso, o fato de uma confirmação ter sido recebida na máquina do usuário não quer dizer necessariamente que o host remoto funcionou por tempo suficiente para atualizar o banco de dados. Uma confirmação fim a fim verdadeira, cujo recebimento indica que o trabalho foi realmente realizado e cuja falta indica que ele não foi cumprido, talvez seja algo impossível de alcançar. Esse assunto é discutido com mais detalhes por Saltzer et al. (1984).

6.3 Controle de congestionamento

Se as entidades de transporte em muitas máquinas enviarem muitos pacotes para a rede com muita rapidez, a rede ficará congestionada, com o desempenho degradado enquanto os pacotes são atrasados e perdidos. Controlar o congestionamento para evitar esse problema é responsabilidade conjunta das camadas de rede e transporte. O congestionamento ocorre nos roteadores, de modo que é detectado na camada de rede. Porém, o congestionamento por fim é causado pelo tráfego enviado para a rede pela camada de transporte. O único modo eficaz de controlar o congestionamento é fazer com que os protocolos de transporte enviem pacotes mais lentamente para a rede.

No Capítulo 5, estudamos os mecanismos de controle de congestionamento na camada de rede. Nesta seção, estudaremos a outra metade do problema, os mecanismos de controle de congestionamento na camada de transporte. Depois de descrever os objetivos do controle de congestionamento, descreveremos como os hosts podem regular a taxa com que enviam pacotes para a rede. A Internet conta bastante com a camada de transporte para o controle de congestionamento, e algoritmos específicos são elaborados para TCP e outros protocolos.

6.3.1 Alocação desejável de largura de banda

Antes de descrevermos como regular o tráfego, temos de entender o que estamos tentando alcançar executando um algoritmo de controle de congestionamento. Ou seja, temos de especificar o estado em que um bom algoritmo de controle de congestionamento operará na rede. O objetivo é mais do que simplesmente evitar o congestionamento. É encontrar uma boa alocação de largura de banda para as entidades de transporte que estão usando a rede. Uma boa alocação oferecerá bom desempenho, pois usa toda a largura de banda disponível mas evita congestionamento, será justa entre entidades de transporte concorrentes e rastreará rapidamente as mudanças nas demandas de tráfego. Vamos esclarecer cada um desses critérios por vez.

Eficiência e potência

Uma alocação eficiente de largura de banda por entidades de transporte usará toda a capacidade da rede que se encontra disponível. Porém, não é muito certo pensar que, se existe um enlace de 100 Mbps, cinco entidades de transporte deverão receber 20 Mbps cada uma. Elas normalmente deverão receber menos de 20 Mbps para que tenham um bom desempenho. O motivo é que o tráfego normalmente é feito por rajada. Lembre-se de que, na Seção 5.3, descrevemos o **goodput** (ou vazão normalizada, a taxa de pacotes úteis que chegam ao receptor) como uma função da carga oferecida. Essa curva e uma curva correspondente para o atraso como uma função da carga oferecida são apresentadas na Figura 6.16.

À medida que a carga aumenta na Figura 6.16(a), o goodput inicialmente aumenta na mesma velocidade, mas quando a carga se aproxima da capacidade, o goodput aumenta mais gradualmente. Isso ocorre porque as rajadas de tráfego ocasionalmente podem se acumular e causar mais perdas nos buffers dentro da rede. Se o protocolo de transporte for mal projetado e retransmitir pacotes que foram atrasados mas não perdidos, a rede pode entrar em colapso de congestionamento. Nesse estado, os transmissores estão furiosamente enviando pacotes, mas cada vez menos trabalho útil está sendo realizado.

O atraso correspondente é dado na Figura 6.16(b). Inicialmente, o atraso é fixo, representando o atraso de propagação pela rede. À medida que a carga se aproxima da capacidade, o atraso aumenta, lentamente a princípio e depois muito mais rapidamente. Isso novamente é por causa do tráfego que tende a se acumular em carga alta. O atraso não pode realmente ir até o infinito, exceto em um modelo em que os roteadores possuem buffers infinitos. Em vez disso, os pacotes serão perdidos após experimentarem um atraso máximo de buffering.

Para o goodput e o atraso, o desempenho começa a degradar no início do congestionamento. Intuitivamente, obteremos o melhor desempenho a partir da rede se alocarmos largura de banda até que o atraso comece a cair rapidamente. Esse ponto está abaixo da capacidade. Para identificá-lo, Kleinrock (1979) propôs a métrica da **potência**, onde

$$potência = \frac{carga}{atraso}$$

A potência inicialmente aumentará com a carga oferecida, pois o atraso continua sendo pequeno e relativamente constante, mas alcançará um máximo e cairá à medida

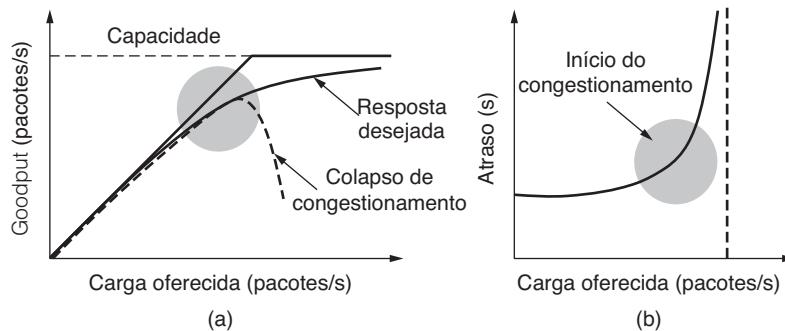



Figura 6.16 | (a) Goodput e (b) atraso como função da carga oferecida.

que o atraso crescer rapidamente. A carga com a mais alta potência representa uma carga eficiente para a entidade de transporte colocar sobre a rede.

Imparcialidade max-min

Na discussão anterior, não falamos sobre como dividir a largura de banda entre diferentes transmissores de transporte. Isso parece uma questão simples de responder — damos a todos os transmissores uma fração igual da largura de banda —, mas ela envolve várias considerações.

Talvez a primeira consideração seja perguntar o que esse problema tem a ver com o controle de congestionamento. Afinal, se a rede der a um transmissor alguma quantidade de largura de banda para usar, o transmissor deve usar apenas essa largura de banda. Porém, normalmente acontece que as redes não possuem uma reserva de largura de banda estrita para cada fluxo ou conexão. Isso pode acontecer para alguns fluxos, se a qualidade de serviço for admitida, mas muitas conexões buscarão usar qualquer largura de banda que estiver disponível ou serão reunidas pela rede sob uma alocação comum. Por exemplo, os serviços diferenciados da IETF separam o tráfego em duas classes e as conexões competem pela largura de banda dentro de cada classe. Os roteadores IP normalmente têm todas as conexões competindo pela mesma largura de banda. Nessa situação, é o mecanismo de controle de congestionamento que está alocando largura de banda às conexões concorrentes.

Uma segunda consideração é o que significa uma fatia justa para os fluxos em uma rede. Seria muito simples se *N* fluxos usassem um único enlace, quando todos eles poderiam ter 1/*N* da largura de banda (embora a eficiência dite que eles usem pouco menos se o tráfego for em rajadas). Mas o que acontece se os fluxos tiverem caminhos de rede diferentes, porém sobrepostos? Por exemplo, um fluxo pode atravessar três enlaces e os outros fluxos podem atravessar um enlace. O fluxo de três enlaces consome mais recursos da rede. Pode ser mais justo em certo sentido dar-lhe menos largura de banda do que os fluxos de um enlace. Certamente, deve ser possível dar suporte a mais fluxos de um enlace reduzindo a largura de banda do fluxo de três enlaces. Esse ponto demonstra uma tensão inerente entre justiça e eficiência.

Porém, adotaremos uma noção de justiça que não depende do comprimento do caminho da rede. Até mesmo com esse modelo simples, dar às conexões uma fração igual da largura de banda é um pouco complicado, pois diferentes conexões usarão diferentes caminhos pela rede e esses caminhos por si só terão diferentes capacidades. Nesse caso, é possível que um fluxo seja engarrafado em um enlace mais adiante e use uma parte menor de um enlace anterior do que outros fluxos; reduzir a largura de banda dos outros fluxos causaria lentidão, mas não ajudaria o fluxo com gargalo.

A forma de justiça que normalmente é desejada para uso da rede é a **imparcialidade max-min**. Uma alocação é imparcial max-min se a largura de banda dada a um fluxo não puder ser aumentada sem diminuir a largura de banda dada a outro fluxo com uma alocação que não seja maior. Ou seja, aumentar a largura de banda de um fluxo só tornará a situação pior para fluxos menos afortunados.

Vejamos um exemplo. Um alocação imparcial max-min é mostrada para uma rede com quatro fluxos, *A*, *B*, *C* e *D*, na Figura 6.17. Cada um dos enlaces entre os roteadores tem a mesma capacidade, considerada como 1 unidade, embora no caso geral os enlaces tenham diferentes capacidades. Três fluxos competem pelo enlace inferior esquerdo entre as rotas *R4* e *R5*. Cada um desses fluxos, portanto, recebe 1/3 do enlace. O fluxo restante, *A*, compete com *B* no enlace de *R2* a *R3*. Como *B* tem uma alocação de 1/3, *A* recebe os 2/3 restantes do enlace. Observe que todos os outros enlaces possuem capacidade de reserva. Porém, essa capacidade não pode ser dada a qualquer um dos fluxos sem diminuir a capacidade de outro fluxo mais baixo. Por exemplo, se mais da largura de banda do enlace entre *R2* e *R3* for dada ao fluxo *B*, haverá menos para o fluxo *A*. Isso é razoável, pois o fluxo *A* já tem mais largura de banda. Porém, a capacidade de fluxo *C* ou *D* (ou ambos) deve ser diminuída para oferecer mais largura de banda a *B*, e esses

fluxos terão menos largura de banda do que *B*. Assim, a alocação é imparcial max-min.

Alocações max-min podem ser calculadas dado um conhecimento global da rede. Um modo intuitivo de pensar sobre elas é imaginar que a taxa para todos os fluxos começa com zero e é aumentada lentamente. Quando a taxa alcança um gargalo para qualquer fluxo, então esse fluxo deixa de aumentar. Todos os outros fluxos continuam a aumentar, compartilhando igualmente a capacidade disponível, até que eles também alcancem seus respectivos gargalos.

Uma terceira consideração é o nível para avaliar a imparcialidade. Uma rede poderia ser imparcial no nível de conexões, conexões entre um par de hosts ou todas as conexões por host. Examinamos essa questão quando discutimos WFQ (Weighted Fair Queueing) na Seção 5.4 e concluímos que cada uma dessas definições tem seus problemas. Por exemplo, a definição de imparcialidade por host significa que um servidor ocupado não se sai melhor que um telefone móvel, embora a definição de imparcialidade por conexão encoraje os hosts a abrir mais conexões. Como não existe uma resposta clara, a imparcialidade normalmente é considerada por conexão, mas a imparcialidade precisa geralmente não é um problema. É mais importante na prática que nenhuma conexão tenha falta de largura de banda do que as demais conexões que recebem precisamente a mesma quantidade de largura de banda. De fato, com o TCP, é possível abrir várias conexões e competir pela largura de banda com mais agressividade. Essa tática é usada por aplicações famintas por largura de banda, como BitTorrent para compartilhamento de arquivos peer-to-peer.

Convergência

Um último critério é que o algoritmo de controle de congestionamento convirja rapidamente para uma alocação imparcial e eficiente da largura de banda. Essa discussão do ponto de operação desejável considera um ambiente de rede estático. Porém, as conexões sempre estão indo e vindo em uma rede, e a largura de banda necessária para determinada conexão também variará com o tempo, por exemplo, à medida que um usuário navega pelas páginas Web e ocasionalmente baixa vídeos grandes.

Devido à variação na demanda, o ponto de operação ideal para a rede varia com o tempo. Um bom algoritmo de controle de congestionamento rapidamente converge para o ponto de operação ideal, e deve acompanhar esse ponto à medida que ele muda com o tempo. Se a convergência for muito lenta, o algoritmo nunca estará próximo do ponto de operação em mudança. Se o algoritmo não for estável, ele pode deixar de convergir para o ponto certo em alguns casos, ou até mesmo oscilar em torno do ponto certo.

Um exemplo de alocação de largura de banda que muda com o tempo e converge rapidamente aparece na Figura 6.18. Inicialmente, o fluxo 1 tem toda a largura de banda. Um segundo depois, o fluxo 2 começa. Ele também precisa de largura de banda. A alocação rapidamente muda para dar a cada um desses fluxos metade da largura de banda. Em 4 segundos, um terceiro fluxo se junta. Porém, esse fluxo usa apenas 20 por cento da largura de banda, que é menos do que sua fatia imparcial (que é um terço). Os fluxos 1 e 2 rapidamente se ajustam, dividindo a largura de banda disponível para que cada um tenha 40 por cento da largura de banda. Em 9 segundos, o segundo fluxo sai e o terceiro fluxo permanece inalterado. O primeiro fluxo rapidamente captura 80 por cento da largura de banda. Em todos os momentos, a largura de banda alocada total é aproximadamente 100 por cento, de modo que a rede é totalmente usada, e os fluxos concorrentes recebem o mesmo tratamento (mas não têm que usar mais largura de banda do que precisam).

6.3.2 Regulando a velocidade de envio

Agora é hora do curso principal. Como regulamos as taxas de envio para obter uma alocação de largura de banda desejável? A taxa de envio pode ser limitada por dois fatores. O primeiro é o controle de fluxo, caso exista um buffering insuficiente no receptor. O segundo é o congestionamento, caso exista capacidade insuficiente na rede. Na Figura 6.19, vemos esse problema ilustrado de forma hidráulica. Na Figura 6.19(a), vemos um cano grosso levando a um receptor de pequena capacidade. Essa é uma situação limitada de controle de fluxo. Desde que o transmissor não envie mais água do que o balde pode conter, água não será perdida. Na Figura 6.19(b), o fator limitador não é a capacidade do

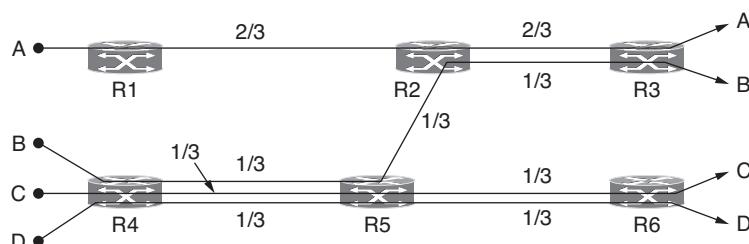



Figura 6.17 | Alocação de largura de banda max-min para quatro fluxos.

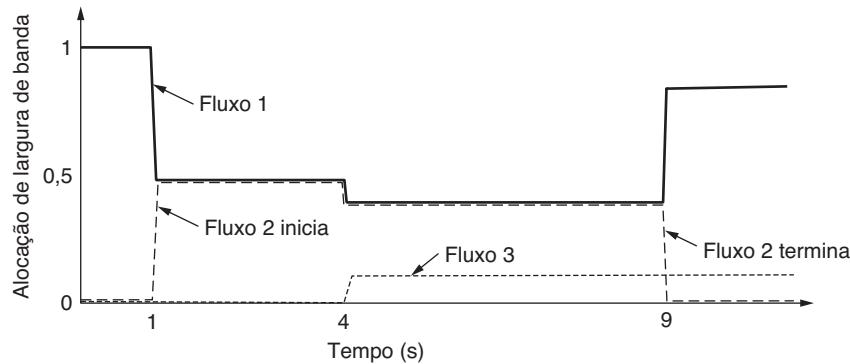



Figura 6.18 | Mudando a alocação de largura de banda com o tempo.

balde, mas a capacidade interna de transporte da rede. Se muita água chegar rapidamente, ela se acumulará e parte será perdida (nesse caso, estourando a capacidade do funil).

Esses casos podem parecer semelhantes ao do transmissor, de modo que transmitir muito rápido faz com que os pacotes sejam perdidos. Porém, eles têm diferentes causas e exigem diferentes soluções. Já falamos sobre uma solução de controle de fluxo com uma janela de tamanho variável. Agora, consideraremos uma solução de controle de congestionamento. Como um desses problemas pode ocorrer, o protocolo de transporte em geral precisará executar as duas soluções e diminuir a velocidade se houver algum problema.

O modo como um protocolo de transporte deve regular a velocidade de envio depende da forma do feedback retornado pela rede. Diferentes camadas de rede podem retornar diferentes tipos de feedback. O feedback pode ser explícito ou implícito, e pode ser preciso ou impreciso.

Um exemplo de projeto explícito e preciso é quando os roteadores dizem aos transmissores a taxa em que podem transmitir. Os projetos na literatura, como o XCP (eXplicit Congestion Protocol), operam dessa maneira (Katabi et al., 2002). Um projeto explícito e impreciso é o uso de ECN (Explicit Congestion Notification) com TCP. Nesse projeto, os roteadores definem bits nos pacotes que sofrem congestionamento para alertar os transmissores para reduzir a velocidade, mas eles não lhes informam quanto devem reduzir.

Em outros projetos, não existe sinal explícito. O FAST TCP mede o atraso de ciclo e usa essa métrica como um

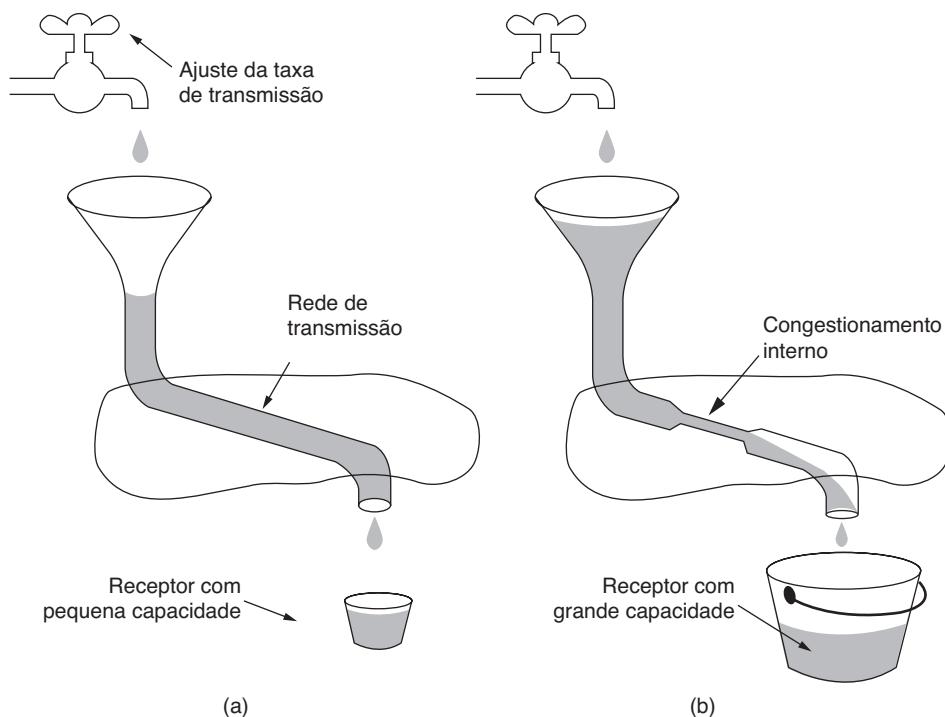



Figura 6.19 | (a) Uma rede rápida alimentando um receptor de baixa capacidade. (b) Uma rede lenta alimentando um receptor de alta capacidade.

sinal para evitar congestionamento (Wei et al., 2006). Finalmente, na forma de controle de congestionamento mais prevalente hoje na Internet, o TCP com tail drop ou roteadores RED, a perda de pacotes é deduzida e usada para sinalizar que a rede ficou congestionada. Existem muitas variantes dessa forma de TCP, incluindo CUBIC TCP, que é usado no Linux (Ha et al., 2008). Certas combinações também são possíveis. Por exemplo, o Windows inclui TCP composto, que usa perda de pacotes e atraso como sinais de feedback (Tan e outros, 2006). Esses projetos são resumidos na Tabela 6.3.

Se for dado um sinal explícito e preciso, a entidade de transporte pode usar esse sinal para ajustar sua taxa ao novo ponto de operação. Por exemplo, se o XCP disser aos transmissores que velocidade deve ser usada, estes podem simplesmente usar essa velocidade. Nos outros casos, porém, alguma tentativa será feita aleatoriamente. Na ausência de um sinal de congestionamento, os transmissores deverão aumentar sua velocidade. Quando um sinal de congestionamento é dado, os transmissores devem diminuir sua velocidade. O modo como a velocidade é aumentada ou diminuída é dado por uma **lei de controle**. Essas leis possuem um efeito importante sobre o desempenho.

Chiu e Jain (1989) estudaram o caso do feedback de congestionamento binário e concluíram que **AIMD** (**Additive Increase Multiplicative Decrease**) é a lei de controle apropriada para chegar ao ponto operacional eficiente e imparcial. Para defender esse caso, eles construíram um argumento gráfico para o caso simples de duas conexões competindo pela largura de banda de um único enlace. O grafo na Figura 6.20 mostra a largura de banda alocada ao usuário 1 no eixo x e ao usuário 2 no eixo y. Quando a alocação é imparcial, os dois usuários recebem a mesma quantidade de largura de banda. Isso pode ser visto pela linha de imparcialidade pontilhada. Quando as alocações somam 100 por cento da capacidade do enlace, a alocação é eficiente. Isso pode ser visto pela linha de eficiência pontilhada. Um sinal de congestionamento é dado pela rede para os dois usuários quando a soma de suas alocações

Protocolo	Sinal	Explícito?	Preciso?
XCP	Velocidade a usar	Sim	Sim
XCP com ECN	Advertência de congestionamento	Sim	Não
FAST TCP	Atraso fim a fim	Não	Sim
TCP COMPOSTO	Perda de pacote e atraso fim a fim	Não	Sim
CUBIC TCP	Perda de pacote	Não	Não
TCP	Perda de pacote	Não	Não

Tabela 6.3 | Sinais de alguns protocolos de controle de congestionamento.

cruza essa linha. A interseção dessas linhas é o ponto operacional desejado, quando os dois usuários têm a mesma largura de banda e toda a largura de banda da rede é usada.

Considere o que acontece a partir de alguma alocação inicial se o usuário 1 e o usuário 2 aumentarem aditivamente suas respectivas larguras de banda com o tempo. Por exemplo, os usuários podem cada um aumentar sua velocidade de envio em 1 Mbps a cada segundo. Por fim, o ponto operacional cruza a linha da eficiência e os dois usuários recebem um sinal de congestionamento da rede. Nesse estágio, eles precisam reduzir suas alocações. Porém, uma diminuição aditiva simplesmente faria com que eles osciassem ao longo de uma linha aditiva. Essa situação pode ser vista na Figura 6.20. O comportamento manterá o ponto operacional próximo da eficiência, mas não necessariamente será imparcial.

De modo semelhante, considere o caso em que os dois usuários aumentam multiplicativamente sua largura de banda com o tempo até que recebam um sinal de congestionamento. Por exemplo, os usuários podem aumentar sua velocidade de envio em 10 por cento a cada segundo. Se eles então diminuírem multiplicativamente sua velocidade de transmissão, o ponto operacional dos usuários simplesmente oscilará ao longo de uma linha multiplicativa. Esse comportamento também aparece na Figura 6.20. A linha multiplicativa tem uma inclinação diferente da linha aditiva. (Ela aponta para a origem, enquanto a linha aditiva tem um ângulo de 45 graus.) Porém, de outras maneiras, ela não é melhor. Em nenhum dos casos os usuários convergirão para as velocidades de envio ótimas, que sejam imparciais e eficientes.

Agora, considere o caso em que os usuários aumentam aditivamente suas alocações de largura de banda e depois as diminuem multiplicativamente quando o congestionamento é sinalizado. Esse comportamento é a lei de controle AIMD, e aparece na Figura 6.21. Pode-se ver que o caminho traçado por esse comportamento converge para o ponto ótimo, que é imparcial e eficiente. Essa convergência acontece independentemente do ponto de partida, tornando o AIMD muito útil. Por esse mesmo argumento, a única outra combinação, o aumento multiplicativo com diminuição aditiva, divergiria do ponto ótimo.

AIMD é a lei de controle usada pelo TCP, com base nesse argumento e em outro argumento de estabilidade (o de que é fácil levar a rede ao congestionamento e difícil recuperar, de modo que a política de aumento deve ser suave e a política de diminuição, agressiva). Ele não é muito imparcial, pois as conexões TCP ajustam seu tamanho de janela por determinado valor a cada tempo de ciclo. Diferentes conexões terão diferentes tempos de ciclo. Isso leva a uma imparcialidade na qual as conexões com hosts mais próximos receberão mais largura de banda do que as conexões com hosts distantes, tudo o mais sendo igual.

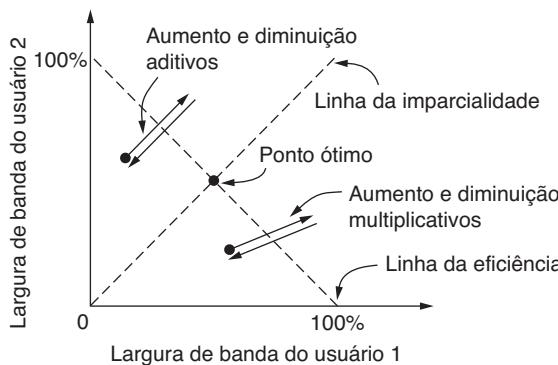



Figura 6.20 | Ajustes de largura de banda aditiva e multiplicativa.

Na Seção 6.5, descreveremos com detalhes como o TCP implementa uma lei de controle AIMD para ajustar a velocidade de envio e oferece controle de congestionamento. Essa tarefa é mais difícil do que parece, pois as velocidades são medidas por algum intervalo e o tráfego é feito em rajadas. Em vez de ajustar a velocidade diretamente, uma estratégia normalmente utilizada na prática é ajustar o tamanho de uma janela deslizante. O TCP usa essa estratégia. Se o tamanho da janela é *W* e o tempo de ciclo é *RTT*, a taxa equivalente é *W*/*RTT*. Essa estratégia é fácil de combinar com controle de fluxo, que já usa uma janela, e tem a vantagem de que o transmissor regula os pacotes usando confirmações e, portanto, atrasa em um *RTT* se parar de receber relatórios de que os pacotes estão saindo da rede.

Como uma última questão, pode haver muitos protocolos de transporte diferentes que enviam tráfego para a rede. O que acontecerá se os diferentes protocolos competirem com diferentes leis de controle para evitar o congestionamento? Alocações de largura de banda desiguais, isso é o que acontece. Como o TCP é a forma dominante de controle de congestionamento na Internet, existe pressão significativa da comunidade por novos protocolos de transporte a ser projetados, de modo que concorram de forma justa com ele. Os primeiros protocolos de streaming de mídia causavam problemas reduzindo excessivamente a utilização do TCP no controle de congestionamento, pois não competiam com imparcialidade. Isso trouxe a ideia de controle de congestionamento **com um TCP amigável**, em que o TCP e protocolos de transporte diferentes podem ser livremente mesclados sem efeitos prejudiciais (Floyd et al., 2000).

6.3.3 Problemas da rede sem fios

Os protocolos de transporte como o TCP, que implementam controle de congestionamento, devem ser independentes das tecnologias subjacentes das camadas de rede e de enlace. Essa é uma boa teoria, mas na prática existem problemas com redes sem fio. O principal deles é que a perda de pacotes normalmente é usada como um sinal de congestionamento, inclusive pelo TCP, conforme já discutimos. As redes sem fio perdem pacotes o tempo todo, devido a erros de transmissão.

Com a lei de controle AIMD, a alta vazão requer níveis muito pequenos de perda de pacotes. As análises de Padhye et al. (1998) mostram que a vazão se relaciona com o inverso da raiz quadrada da taxa de perda de pacotes. Na prática, isso significa que a taxa de perdas para conexões TCP velozes é muito pequena; 1 por cento é uma taxa de perdas moderada, e, quando a taxa atinge 10 por cento, a conexão efetivamente terá deixado de funcionar. Porém, para redes sem fio como as LANs 802.11, taxas de perda de quadros de pelo menos 10 por cento são comuns. Essa diferença significa que, sem medidas de proteção, os esquemas de controle de congestionamento que usam perda de pacotes como um sinal desnecessariamente sufocarão as conexões que atuam sobre enlaces sem fio em velocidades muito baixas.

Para funcionar bem, as únicas perdas de pacote que o algoritmo de controle de congestionamento deve observar são perdas devidas à largura de banda insuficiente, e não perdas devidas a erros de transmissão. Uma solução para esse problema é mascarar as perdas sem fios usando retransmissões pelo enlace sem fios. Por exemplo, o 802.11 usa um protocolo stop-and-wait para entregar cada qua-

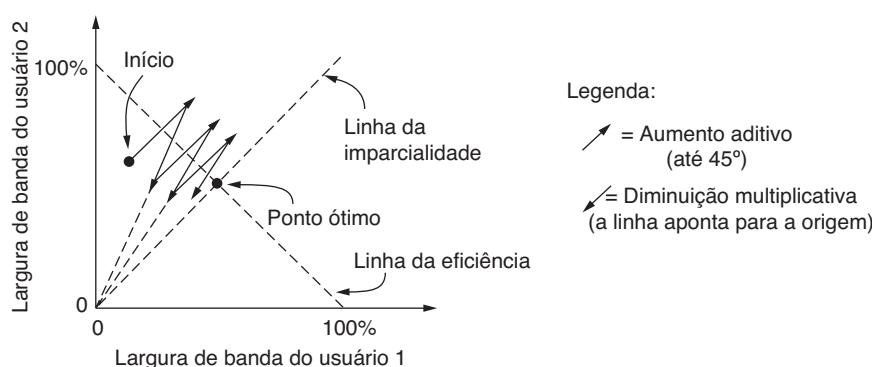



Figura 6.21 | Lei de controle do aumento aditivo com diminuição multiplicativa (AIMD).

dro, tentando realizar transmissões novamente várias vezes, se for preciso, antes de informar uma perda de pacote à camada superior. No caso normal, cada pacote é entregue apesar de erros de transmissão transitórios, que não são visíveis às camadas mais altas.

A Figura 6.22 mostra um caminho com um enlace com fios e sem fios, para o qual a estratégia de máscara é utilizada. Existem dois aspectos a observar. Primeiro, o transmissor não necessariamente sabe que o caminho inclui um enlace sem fios, pois tudo o que ele vê é o enlace com fios ao qual está conectado. Os caminhos na Internet são heterogêneos e não existe um método geral para que o transmissor saiba quais tipos de enlaces compreendem o caminho inteiro. Isso complica o problema do controle de congestionamento, pois não há um modo fácil de usar um protocolo para enlaces sem fios e outro protocolo para enlaces com fios.

O segundo aspecto é um quebra-cabeça. A figura mostra dois mecanismos que são controlados por perda: retransmissões de quadro da camada de enlace e controle de congestionamento na camada de transporte. O quebra-cabeça é como esses dois mecanismos podem coexistir sem se confundirem. Afinal, uma perda deve fazer com que apenas um mecanismo tome alguma ação, pois esse é um erro de transmissão ou um sinal de congestionamento. Não é possível que haja ambos. Se os dois mecanismos tomarem alguma ação (retransmitindo o quadro e diminuindo a velocidade de envio), então voltaremos ao problema original de transportes que rodam mais lentamente por enlaces sem fios. Considere esse quebra-cabeça por um instante e veja se você pode solucioná-lo.

A solução é que os dois mecanismos atuam em escalas de tempo diferentes. As retransmissões da camada de enlace acontecem na ordem de microssegundos a milissegundos para enlaces sem fios, como o 802.11. Os timers de perda nos protocolos de transporte disparam na ordem de milissegundos a segundos. A diferença é de três ordens de grandeza. Isso permite que os enlaces sem fios detectem as perdas de quadros e retransmitam os quadros para reparar erros de transmissão muito tempo antes que a perda de pacote seja deduzida pela entidade de transporte.

A estratégia de máscara é suficiente para permitir que os protocolos de transporte funcionem bem pela maioria dos enlaces sem fios. Porém, essa nem sempre é uma solução ideal. Alguns enlaces sem fios possuem longos tempos de ciclo, como os satélites. Para esses enlaces, outras técnicas devem ser usadas para mascarar a perda, como FEC (Forward Error Correction), ou o protocolo de transporte deve usar um sinal sem perda para o controle de congestionamento.

Um segundo problema do controle de congestionamento por enlaces sem fios é a capacidade variável. Ou seja, a capacidade de um enlace sem fios mudar com o tempo, às vezes bruscamente, à medida que os nós se movem e a relação sinal-ruído varia com a mudança das condições no canal. Isso é diferente dos enlaces com fios, cuja capacidade é fixa. O protocolo de transporte precisa se adaptar à capacidade variável dos enlaces sem fios, ou então congestionará a rede ou deixará de usar a capacidade disponível.

Uma solução possível para esse problema é simplesmente não se preocupar com isso. Essa estratégia é viável porque os algoritmos de controle de congestionamento já deverão tratar do caso de novos usuários entrando na rede ou usuários existentes mudando sua velocidade de envio. Embora a capacidade dos enlaces com fios seja fixa, o comportamento variável de outros usuários se apresenta como variabilidade na largura de banda que está disponível a determinado usuário. Assim, é possível simplesmente usar TCP por um caminho com um enlace sem fios 802.11 e obter um desempenho razoável.

Porém, quando existe muita variabilidade sem fios, os protocolos de transporte projetados para enlaces com fios podem ter dificuldade de acompanhar, e oferecem um desempenho fraco. A solução nesse caso é um protocolo de transporte que seja projetado para enlaces com fios. Um ambiente particularmente desafiador é uma rede em malha sem fios em que vários enlaces sem fios interferindo se cruzam, e as rotas mudam devido à mobilidade e existe muita perda. Existe pesquisa em andamento nessa área. Consulte em Li et al. (2009) um exemplo de projeto de protocolo de transporte sem fios.

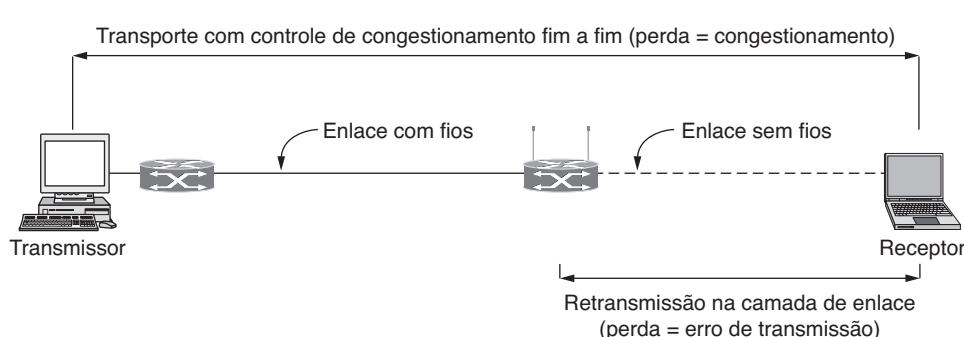



Figura 6.22 | Controle de congestionamento sobre um caminho com um enlace sem fios.

6.4 Os protocolos de transporte da Internet: UDP

A Internet tem dois protocolos principais na camada de transporte, um protocolo não orientado a conexões e outro orientado a conexões. Eles complementam um ao outro. O protocolo não orientado a conexões é o UDP. Ele faz quase tudo além de enviar pacotes entre aplicações, permitindo que as aplicações criem seus próprios protocolos em cima, conforme a necessidade. O protocolo orientado a conexões é o TCP. Ele faz quase tudo. Faz conexões e acrescenta confiabilidade com retransmissões, junto com controle de fluxo e controle de congestionamento, tudo em favor das aplicações que o utilizam.

Nas próximas seções, estudaremos UDP e TCP. Começaremos com UDP, pois é o mais simples. Também veremos dois usos do UDP. Como UDP é um protocolo da camada de transporte que normalmente é executado no sistema operacional e os protocolos que usam UDP normalmente são executados no espaço do usuário, esses usos poderiam ser considerados aplicações. Porém, as técnicas que eles empregam são úteis para muitas aplicações, e o melhor é considerá-los pertencentes a um serviço de transporte, e por isso serão explicados aqui.

6.4.1 Introdução ao UDP

O conjunto de protocolos da Internet admite um protocolo de transporte não orientado a conexões, o protocolo de datagrama do usuário, ou **UDP** (**User Datagram Protocol**). O UDP oferece um meio para as aplicações enviarem datagramas IP encapsulados sem que seja necessário estabelecer uma conexão. O UDP é descrito na RFC 768.

O UDP transmite **segmentos** que consistem em um cabeçalho de 8 bytes, seguido pela carga útil. O cabeçalho é mostrado na Figura 6.23. As duas **portas** servem para identificar os pontos extremos nas máquinas de origem e destino. Quando um pacote UDP chega, sua carga útil é entregue ao processo associado à porta de destino. Essa associação ocorre quando a primitiva BIND ou algo semelhante são usados, como vimos no Quadro 6.1 para o TCP (o processo de vinculação é idêntico para o UDP). Pense nas portas como caixas de correio que as aplicações podem utilizar para receber pacotes. Falaremos mais sobre elas quando descrevermos o TCP, que também usa portas. De fato, o principal valor de ter o UDP em relação ao uso do IP bruto é a adição das portas de origem e destino. Sem os campos portas, a camada de transporte não saberia o que fazer com o pacote que chega. Com eles, a camada entrega o segmento encapsulado à aplicação correta.

A porta de origem é necessária principalmente quando uma resposta precisa ser enviada de volta à origem. Copiando o campo *Porta de origem* do segmento de entrada no campo *Porta de destino* do segmento de saída, o processo que transmite a resposta pode especificar qual processo na máquina transmissora deve recebê-lo.

O campo *Comprimento do UDP* inclui o cabeçalho de 8 bytes e os dados. O comprimento mínimo é de 8 bytes, para incluir o cabeçalho. O comprimento máximo é de 65.515 bytes, que é menor que o maior número que caberá em 16 bits, devido ao limite de tamanho nos pacotes IP.

Um campo opcional de *Checksum do UDP* também é fornecido para gerar confiabilidade extra. Ele faz o checksum do cabeçalho, dos dados e de um pseudocabeçalho conceitual do IP. Ao realizar um cálculo, o campo de *Checksum* é definido como zero e o campo de dados é preenchido com um byte zero adicional se seu comprimento for um número ímpar. O algoritmo de checksum consiste simplesmente em somar todas as palavras de 16 bits com complemento de um e apanhar o complemento de um da soma. Por conseguinte, quando o receptor realiza o cálculo sobre o segmento inteiro, incluindo o campo de *Checksum*, o resultado deve ser 0. Se o checksum não for calculado, ele será armazenado como 0, pois, por uma feliz coincidência da aritmética de complemento de um, um valor 0 verdadeiro calculado é armazenado com todos os bits iguais a 1. É tolice desativá-lo, a menos que a qualidade dos dados não tenha importância (por exemplo, no caso de voz digitalizada).

O pseudocabeçalho para o caso do IPv4 aparece na Figura 6.24. Ele contém os endereços IPv4 de 32 bits das máquinas de origem e destino, o número de protocolo para o UDP (17) e a contagem de bytes para o segmento UDP (incluindo o cabeçalho). Para o IPv6, ele é diferente, porém similar. A inclusão do pseudocabeçalho no cálculo do checksum do UDP ajuda a detectar pacotes não entregues, mas incluí-lo também infringe a hierarquia de protocolos, pois os endereços IP nele pertencem à camada do IP, e não à camada do UDP. O TCP usa o mesmo pseudocabeçalho para o seu checksum.

Vale a pena mencionar explicitamente algumas ações que o UDP *não* realiza. Ele não realiza controle de fluxo,

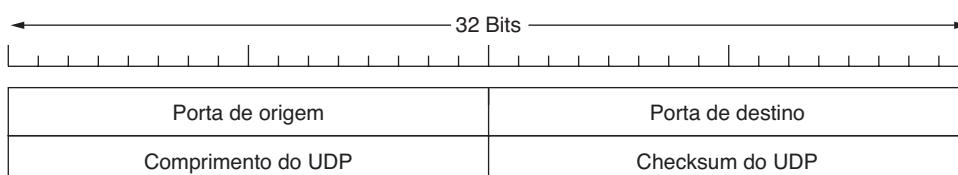



Figura 6.23 | O cabeçalho UDP.

controle de congestionamento ou retransmissão após a recepção de um segmento incorreto. Tudo isso cabe aos processos do usuário. O que ele faz é fornecer uma interface para o protocolo IP com o recurso adicional de demultiplexação de vários processos que utilizam as portas e detecção opcional de erro fim a fim. Isso é tudo que ele faz.

Para aplicações que precisam ter controle preciso sobre o fluxo de pacotes, os erros ou a sincronização, o UDP fornece apenas aquilo que é determinado. Uma área em que ele é especialmente útil é nas situações cliente-servidor. Normalmente, o cliente envia uma solicitação curta para o servidor e espera uma resposta curta de volta. Se a solicitação ou a resposta se perderem, o cliente pode entrar em timeout e tentar novamente. Não apenas o código é simples, mas menos mensagens são necessárias (uma em cada sentido) do que com um protocolo exigindo uma configuração inicial, como o TCP.

Uma aplicação que utiliza o UDP desse modo é o DNS (Domain Name System), que estudaremos no Capítulo 7. Em resumo, um programa que precisa pesquisar o endereço IP de algum nome de host — por exemplo, *www.cs.berkeley.edu* — pode enviar um pacote UDP contendo o nome do host a um servidor DNS. O servidor responde com um pacote UDP que contém o endereço IP do host. Não é necessária nenhuma configuração antecipada e também nenhum encerramento posterior. Basta enviar duas mensagens pela rede.

6.4.2 Chamada de procedimentos remotos (RPC)

Em certo sentido, enviar uma mensagem a um host remoto e obter de volta uma resposta é muito semelhante a criar uma chamada de função em uma linguagem de programação. Em ambos os casos, você começa com um ou mais parâmetros e recebe de volta um resultado. Essa observação levou as pessoas a tentar organizar interações de solicitação/resposta em redes no formato de chamadas de procedimentos. Tal organização torna as aplicações de rede muito mais fáceis de programar e mais familiares. Por exemplo, imagine uma função chamada *get_IP_address*(*host_name*) que funcione enviando um pacote UDP a um servidor DNS e aguardando a resposta, chegando ao timeout e tentando de novo, caso não receba uma resposta com rapidez suficiente. Desse modo, todos os detalhes de redes podem ficar ocultos ao programador.

O trabalho fundamental nessa área foi realizado por Birrell e Nelson (1984). Em resumo, o que Birrell e Nelson sugeriram foi permitir que os programas chamassem procedimentos localizados em hosts remotos. Quando um processo na máquina 1 chama um procedimento na máquina 2, o processo de chamada em 1 é suspenso, e a execução do procedimento chamado ocorre em 2. As informações podem ser transportadas do chamador até o chamado nos parâmetros, e podem voltar no resultado do procedimento. Nenhuma passagem de mensagens é visível para o programador. Essa técnica é conhecida como chamada de procedimento remoto, ou **RPC** (**Remote Procedure Call**), e se tornou a base para muitas aplicações em redes. Tradicionalmente, o procedimento chamador é conhecido como **cliente**, e o procedimento chamado é conhecido como **servidor**; também usaremos esses nomes aqui.

A ideia por trás da RPC é tornar uma chamada de procedimento remoto o mais semelhante possível de uma chamada local. Na forma mais simples, para chamar um procedimento remoto, o programa cliente deve estar vinculado a um pequeno procedimento de biblioteca, chamado **stub do cliente**, que representa o procedimento do servidor no espaço de endereços do cliente. De modo semelhante, o servidor está vinculado a um procedimento chamado **stub do servidor**. Esses procedimentos ocultam o fato de que a chamada de procedimento do cliente até o servidor não é local.

As etapas reais na criação de uma RPC são mostradas na Figura 6.25. A etapa 1 é a chamada do cliente ao stub do cliente. Essa é uma chamada de procedimento local, com os parâmetros colocados na pilha da maneira normal. A etapa 2 é o stub do cliente reunindo os parâmetros em uma mensagem e efetuando uma chamada de sistema para enviar a mensagem. A reunião dos parâmetros é chamado de agrupamento padronizado (marshaling: organizar dados de forma padronizada). A etapa 3 é o sistema operacional enviando a mensagem da máquina cliente até a máquina servidora. A etapa 4 é o sistema operacional passando o pacote recebido ao stub do servidor. Finalmente, a etapa 5 é o stub do servidor chamando o procedimento servidor

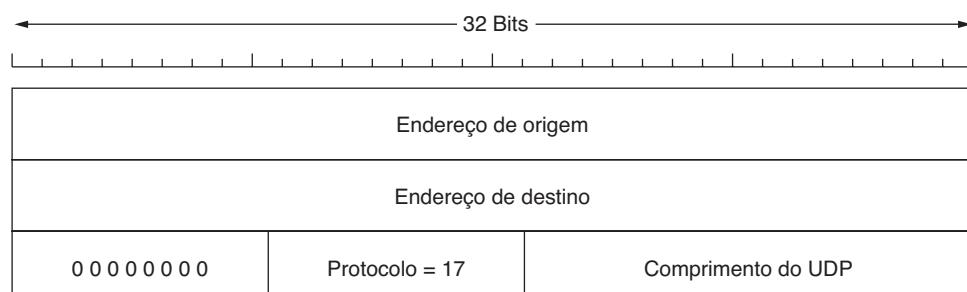



Figura 6.24 | O pseudocabeçalho IPv4 incluído no checksum do UDP.

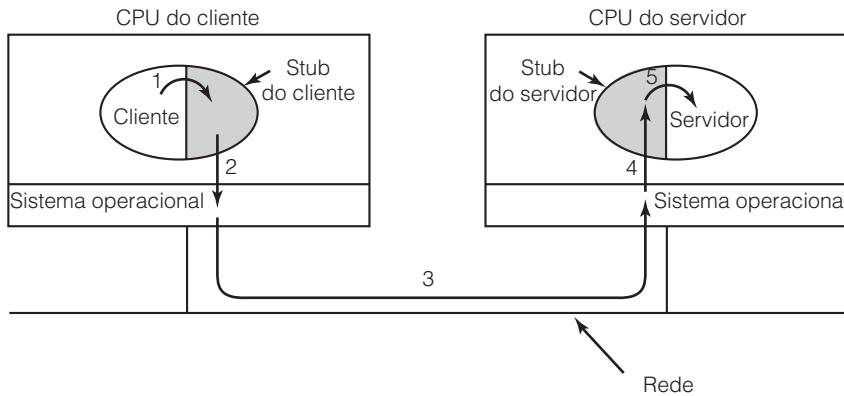



Figura 6.25 | Etapas na criação de uma chamada de procedimento remoto. Os stubs estão sombreados.

com os parâmetros desagrupados. A resposta segue o mesmo caminho no sentido inverso.

O principal detalhe que devemos observar nesse caso é que o procedimento cliente, escrito pelo usuário, simplesmente realiza uma chamada de procedimento normal (isto é, local) ao stub do cliente, que tem o mesmo nome que o procedimento servidor. Tendo em vista que o procedimento cliente e o stub do cliente estão no mesmo espaço de endereços, os parâmetros são repassados no modo habitual. De forma semelhante, o procedimento servidor é chamado por um procedimento em seu espaço de endereços com os parâmetros que espera. Para o procedimento servidor, nada é incomum. Desse modo, em vez de ser realizada uma E/S via soquetes, a comunicação de rede é feita simulando-se uma chamada de procedimento normal.

Apesar da elegância conceitual da RPC, existem algumas fases obscuras. Uma delas é o uso de ponteiros para parâmetros. Normalmente, a passagem de um ponteiro em um procedimento não é problema. O procedimento chamado pode usar um ponteiro do mesmo modo que o chamador o utiliza, porque ambos os procedimentos convivem no mesmo espaço de endereços virtuais. Com a RPC, a passagem de ponteiros é impossível, porque o cliente e o servidor estão em espaços de endereços diferentes.

Em alguns casos, podem ser usados artifícios para tornar possível a passagem de ponteiros. Suponha que o primeiro parâmetro seja um ponteiro para um inteiro *k*. O stub do cliente pode encapsular *k* e enviá-lo para o servidor. O stub do servidor cria então um ponteiro para *k* e o repassa ao procedimento servidor, da maneira esperada. Quando o procedimento servidor devolve o controle ao stub do servidor, este último envia *k* de volta ao cliente, onde o novo *k* é copiado sobre o antigo, pois o servidor pode tê-lo alterado. Na realidade, a sequência de chamada-padrão da chamada por referência foi substituída pela cópia/restauração. Infelizmente, esse artifício nem sempre funciona; por exemplo, se o ponteiro indicar um grafo ou outra estrutura de dados complexa. Por essa razão, algumas restrições devem ser impostas sobre parâmetros para procedimentos chamados remotamente, conforme veremos.

Um segundo problema é que, em linguagens com tipificação fraca, como C, é perfeitamente válido escrever um procedimento que calcula o produto interno de dois vetores (arrays) sem especificar o tamanho de cada vetor. Cada um deles poderia terminar com um valor especial conhecido apenas pelo procedimento de chamada e pelo procedimento chamado. Sob essas circunstâncias, é essencialmente impossível para o stub do cliente encapsular os parâmetros: ele não tem como determinar o tamanho desses parâmetros.

O terceiro é que nem sempre é possível deduzir os tipos de parâmetros, nem mesmo com base em uma especificação formal ou do próprio código. Um exemplo é a função implícita *printf*, que pode ter qualquer número de parâmetros (pelo menos um), e os parâmetros podem ser uma mistura arbitrária de números inteiros, curtos, longos, de caracteres, de strings, de números em ponto flutuante de diversos tamanhos e de outros tipos. Tentar chamar *printf* a partir de um procedimento remoto seria praticamente impossível, porque C é uma linguagem muito permissiva. Porém, uma regra estabelece que a RPC pode ser usada desde que você não use as linguagens C (ou C++), pois estas não são muito usadas pelos desenvolvedores de aplicações distribuídas.

Um quarto problema se relaciona ao uso de variáveis globais. Normalmente, o procedimento chamador e o procedimento chamado podem se comunicar usando variáveis globais, além de parâmetros. Porém, se o procedimento chamado for agora deslocado para uma máquina remota, o código falhará, porque as variáveis globais não serão mais compartilhadas.

Esses problemas não pretendem sugerir que a RPC seja impossível. De fato, ela é amplamente utilizada, mas são necessárias algumas restrições para fazê-la funcionar bem na prática.

Em termos de protocolos da camada de transporte, o UDP é uma boa base para implementar a RPC. Tanto solicitações quanto respostas podem ser enviadas como um úni-

co pacote UDP no caso mais simples, e a operação pode ser rápida. Entretanto, uma implementação também precisa incluir outros mecanismos. Como a solicitação ou a resposta podem se perder, o cliente precisa manter um timer para retransmitir a solicitação. Observe que uma resposta serve como uma confirmação implícita para uma solicitação, de modo que a solicitação não precisa ser confirmada separadamente. Às vezes, os parâmetros ou resultados podem ser maiores que o tamanho máximo de pacote UDP, quando algum protocolo será necessário para entregar mensagens grandes. Se várias solicitações e respostas puderem se sobrepor (como no caso da programação concorrente), um identificador será necessário para combinar a solicitação com a resposta.

Uma preocupação de nível mais alto é que a operação pode não ser idempotente (isto é, não pode ser repetida com segurança). O caso simples é o de operações idempotentes como solicitações e respostas de DNS. O cliente pode seguramente retransmitir essas solicitações várias vezes se nenhuma resposta estiver por vir. Não importa se o servidor nunca recebeu a solicitação, ou se foi a resposta que se perdeu. A resposta, quando finalmente chega, será a mesma (supondo que o banco de dados do DNS não seja atualizado nesse ínterim). Porém, nem todas as operações são idempotentes, por exemplo, quando elas têm efeitos colaterais importantes, como incrementar um timer. A RPC para essas operações exige semântica mais robusta, de modo que, quando o programa chama um procedimento, ele não seja executado várias vezes. Nesse caso, pode ser necessário estabelecer uma conexão TCP e enviar a solicitação por ela, em vez de usar UDP.

6.4.3 Protocolos de transporte em tempo real

A RPC do tipo cliente-servidor é uma área em que o UDP é amplamente utilizado. Outra área é a das aplicações multimídia em tempo real. Em particular, à medida que o rádio da Internet, a telefonia da Internet, a música por demanda, a videoconferência, o vídeo sob demanda e outras aplicações de multimídia se tornaram mais comuns, as pessoas descobriram que cada aplicação estava reinventando aproximadamente o mesmo protocolo de transporte em tempo real. Aos poucos, ficou claro que seria uma boa ideia ter um protocolo de transporte em tempo real genérico para várias aplicações.

Desse modo, foi criado o **RTP** (protocolo de transporte em tempo real, do inglês **Real-time Transport Protocol**). Ele é descrito na RFC 1889 e agora está difundido para aplicações multimídia. Vamos descrever dois aspectos do transporte em tempo real. O primeiro é o protocolo RTP para transportar dados de áudio e vídeo em pacotes. O segundo é o processamento que ocorre, principalmente no receptor, para reproduzir o áudio e o vídeo no momento certo. Essas funções se encaixam na pilha de protocolos, como mostra a Figura 6.26.

O RTP normalmente trabalha no espaço do usuário sobre o UDP (no sistema operacional). Ele opera da maneira descrita a seguir. A aplicação multimídia consiste em vários fluxos de áudio, vídeo, texto e possivelmente outros fluxos. Esses fluxos são armazenados na biblioteca RTP, que se encontra no espaço do usuário, juntamente com a aplicação. Essa biblioteca efetua a multiplexação dos fluxos e os codifica em pacotes RTP, que são então colocados em um soquete. Na outra extremidade do soquete (no sistema operacional), os pacotes UDP são gerados e incorporados a pacotes RTP e entregues ao IP para transmissão por um enlace, como a Ethernet. O processo inverso ocorre no receptor. A aplicação multimídia por fim recebe os dados multimídia da biblioteca RTP. Ela é responsável por reproduzir a mídia. A pilha de protocolos para essa situação é mostrada na Figura 6.26(a). O aninhamento de pacotes é mostrado na Figura 6.26(b).

Como consequência dessa estrutura, é um pouco difícil dizer em que camada o RTP está. Como ele funciona no espaço do usuário e está vinculado ao programa aplicativo, certamente parece ser um protocolo de aplicação. Por outro lado, ele é um protocolo genérico e independente das aplicações que apenas fornecem recursos de transporte, e assim também é semelhante a um protocolo de transporte.

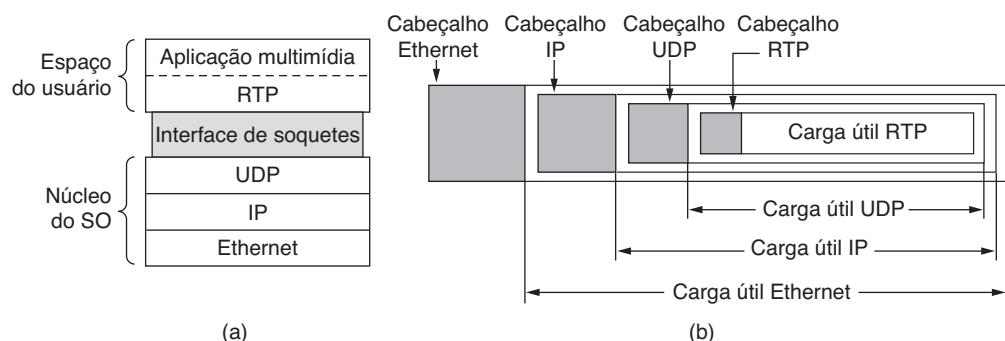



Figura 6.26 | (a) A posição do RTP na pilha de protocolos. (b) O aninhamento de pacotes.

Talvez a melhor descrição do RTP seja como um protocolo de transporte implementado na camada de aplicação, motivo pelo qual está sendo abordado neste capítulo.

RTP — O Protocolo de Transporte em Tempo Real

A função básica do RTP é multiplexar diversos fluxos de dados em tempo real sobre um único fluxo de pacotes UDP. O fluxo UDP pode ser enviado a um único destino (unicasting) ou a vários destinos (multicasting). Como o RTP utiliza simplesmente o UDP normal, seus pacotes não são tratados de maneira especial pelos roteadores, a menos que alguns recursos de qualidade de serviço normais do IP estejam ativos. Em particular, não há nenhuma garantia especial sobre entrega, e pacotes podem ser perdidos, atrasados, adulterados etc.

O formato RTP contém vários recursos para auxiliar os receptores a trabalhar com informações multimídia. Cada pacote enviado em um fluxo RTP recebe um número uma unidade maior que seu predecessor. Essa numeração permite ao destino descobrir se algum pacote está faltando. Se um pacote for omitido, a melhor ação que o destino deve executar fica a cargo da aplicação. Ela pode pular um quadro de vídeo se os pacotes estiverem transportando dados de vídeo, ou aproximar um valor que falta por interpolação, se os pacotes estiverem transportando dados de áudio. A retransmissão não é uma opção prática, pois o pacote retransmitido provavelmente chegaria tarde demais para ser útil. Como consequência, o RTP não tem confirmação e nenhum mecanismo para solicitar retransmissões.

Cada carga útil do RTP pode conter várias amostras, e elas podem ser codificadas de qualquer forma que a aplicação desejar. Para permitir a interoperação, o RTP define vários perfis (por exemplo, um único fluxo de áudio) e, para cada perfil, podem ser permitidos vários formatos de codificação. Por exemplo, um único fluxo de áudio pode ser codificado em amostras PCM de 8 bits a 8 KHz, usando codificação delta, codificação preditiva, codificação GSM, MP3 e assim por diante. O RTP fornece um campo de cabeçalho no qual a origem pode especificar a codificação, mas que não tem nenhum outro envolvimento na maneira de realizar a codificação.

Outro recurso de que muitas aplicações em tempo real necessitam é a marcação com período de tempo. Aqui, a ideia é permitir que a origem associe um período de tempo à primeira amostra em cada pacote. Os períodos de tempo são relativos ao início do fluxo, e assim somente as diferenças entre os períodos de tempo são significativas. Os valores absolutos não têm nenhum significado. Como veremos em breve, esse mecanismo permite ao destino realizar algum buffering e reproduzir cada amostra depois de um número correto de milissegundos, contados desde o início do fluxo, independentemente de quando chegou o pacote contendo a amostra.

O uso de períodos de tempo não apenas reduz os efeitos da variação no atraso da rede, mas também permite a sincronização de vários fluxos. Por exemplo, um programa de televisão digital poderia ter um fluxo de vídeo e dois fluxos de áudio. Os dois fluxos de áudio poderiam ser para broadcasts estereofônicos ou para tratamento de filmes com uma trilha sonora no idioma original e outra dublada no idioma local, dando ao espectador a possibilidade de escolher. Cada fluxo vem de um dispositivo físico diferente, mas se eles forem marcados com um período de tempo baseado em um único timer, poderão ser reproduzidos de modo sincronizado, mesmo que os fluxos sejam transmitidos de maneira um tanto errática.

O cabeçalho do RTP é ilustrado na Figura 6.27. Ele consiste em três palavras de 32 bits e, potencialmente, algumas extensões. A primeira palavra contém o campo de *Versão*, que já está em 2. Vamos esperar que essa versão esteja bem próxima da versão final, pois só falta definir um ponto de código (embora 3 talvez seja definido como uma indicação de que a versão real estava em uma palavra de extensão).

O bit *P* indica que o pacote foi completado até chegar a um múltiplo de 4 bytes. O último byte de preenchimento informa quantos bytes foram acrescentados. O bit *X* indica que um cabeçalho de extensão está presente. O formato e o significado do cabeçalho de extensão não são definidos. O único detalhe definido é que a primeira palavra da extensão fornece o comprimento. Essa é uma válvula de escape para quaisquer exigências imprevistas.

O campo *CC* informa quantas origens de contribuição estão presentes, de 0 a 15 (veja a seguir). O bit *M* é um bit marcador específico da aplicação. Ele pode ser usado para marcar o começo de um quadro de vídeo, o começo de uma palavra em um canal de áudio ou qualquer outro elemento que a aplicação reconheça. O campo *Tipo de carga útil* informa que algoritmo de codificação foi usado (por exemplo, áudio não compactado de 8 bits, MP3 etc.). Tendo em vista que todo pacote apresenta esse campo, a codificação pode mudar durante a transmissão. O campo *Número de sequência* é apenas um timer incrementado em cada pacote RTP enviado. Ele é usado para detectar pacotes perdidos.

O *Período de tempo* é produzido pela origem do fluxo para anotar quando a primeira amostra no pacote foi realizada. Esse valor pode ajudar a reduzir a flutuação de sincronização (chamada jitter) no receptor, desacoplando a reprodução do momento da chegada do pacote. O *Identificador de origem de sincronização* informa a que fluxo o pacote pertence. Esse é o método usado para multiplexar e demultiplexar vários fluxos de dados em um único fluxo de pacotes UDP. Finalmente, o campo *Identificador de origem contribuinte*, se estiver presente, será usado quando houver mixers de áudio no estúdio. Nesse caso, o mixer será a origem de sincronização, e os fluxos que estão sendo misturados serão listados nesse campo.

RTCP — O protocolo de controle de transporte em tempo real

O protocolo RTP tem um irmão caçula, chamado **RTCP** (protocolo de controle de transporte em tempo real,

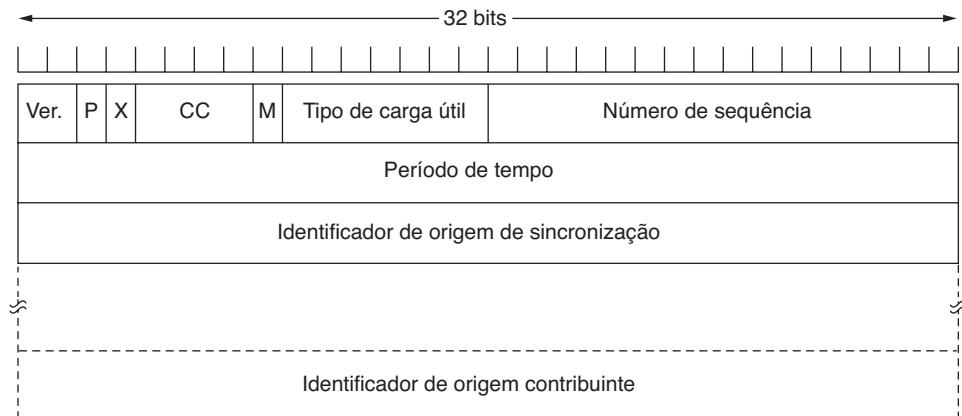



Figura 6.27 | O cabeçalho RTP.

do inglês **Real-time Transport Control Protocol**). Ele é definido com o RTP na RFC 3550 e cuida do feedback, da sincronização e da interface do usuário, mas não transporta nenhuma amostra de mídia.

A primeira função pode ser usada para fornecer feedback sobre o atraso, variação do atraso (ou jitter), largura de banda, congestionamento e outras propriedades de rede para as origens. Essas informações podem ser usadas pelo processo de codificação para aumentar a taxa de dados (e oferecer melhor qualidade) quando a rede estiver funcionando bem e para reduzir a taxa de dados quando houver problemas na rede. Fornecendo feedback contínuo, os algoritmos de codificação podem ser adaptados continuamente para oferecer a melhor qualidade possível sob as circunstâncias atuais. Por exemplo, se a largura de banda aumentar ou diminuir durante a transmissão, a codificação pode passar de MP3 para PCM de 8 bits e para codificação delta, conforme for necessário. O campo *Tipo de carga útil* é usado para informar ao destino qual algoritmo de codificação será empregado no pacote atual, tornando possível variar o algoritmo de acordo com a demanda.

Um problema para fornecer feedback é que os relatórios RTCP são enviados a todos os participantes. Para uma aplicação multicast com um grupo grande, a largura de banda usada pelo RTCP rapidamente se tornaria muito grande. Para impedir que isso aconteça, os transmissores RTCP reduzem a taxa de seus relatórios para consumir coletivamente não mais do que, digamos, 5 por cento da largura de banda de mídia. Para fazer isso, cada participante precisa conhecer a largura de banda de mídia, que ele descobre pelo transmissor, e o número de participantes, que ele estima verificando outros relatórios RTCP.

O RTCP também lida com a sincronização entre fluxos. O problema é que diferentes fluxos podem utilizar clocks distintos, com granularidades e taxas de flutuação diferentes. O RTCP pode ser usado para manter esses elementos sincronizados.

Finalmente, o RTCP fornece um modo para nomear as diversas origens (por exemplo, em texto ASCII). Essas informações podem ser exibidas na tela do receptor, a fim de indicar quem está se comunicando no momento.

Para obter mais informações sobre o RTCP, consulte Perkins (2003).

Transmissão com buffering e controle de jitter

Quando a informação de mídia chega até o receptor, ela precisa ser reproduzida no momento certo. Em geral, esse não será o momento em que o pacote RTP chegou ao receptor, pois os pacotes levarão tempos ligeiramente diferentes para transitar pela rede. Mesmo que os pacotes sejam despachados exatamente com os intervalos certos entre eles no transmissor, eles alcançarão o receptor com tempos relativamente diferentes. Essa variação no atraso é chamada de **jitter**. Até mesmo uma pequena quantidade de jitter no pacote pode causar imperfeições de mídia que a distorcem, como quadros de vídeo irregulares e áudio ininteligível, se a mídia for simplesmente reproduzida quando ela chega.

A solução para esse problema é manter pacotes em **buffer** no receptor antes que sejam reproduzidos, para reduzir o jitter. Como exemplo, na Figura 6.28, vemos um fluxo de pacotes sendo entregue com uma quantidade substancial de jitter. O pacote 1 é enviado do servidor em $t = 0$ s e chega ao cliente em $t = 1$ s. O pacote 2 tem mais atraso e leva 2 s para chegar. Quando um pacote chega, ele é mantido em buffer na máquina cliente.

Em $t = 10$ s, a reprodução começa. Nesse momento, os pacotes de 1 a 6 foram mantidos em buffer, de modo que podem ser removidos do buffer em intervalos uniformes, para gerar uma reprodução suave. No caso geral, não é necessário usar intervalos uniformes, pois os períodos de tempo RTP dizem quando a mídia deve ser reproduzida.

Infelizmente, podemos ver que o pacote 8 está tão atrasado que não está disponível quando entra em cena. Existem duas opções. O pacote 8 pode ser pulado e o player pode prosseguir para os próximos pacotes. Como alternativa, a reprodução pode parar até que o pacote 8 chegue, crian-

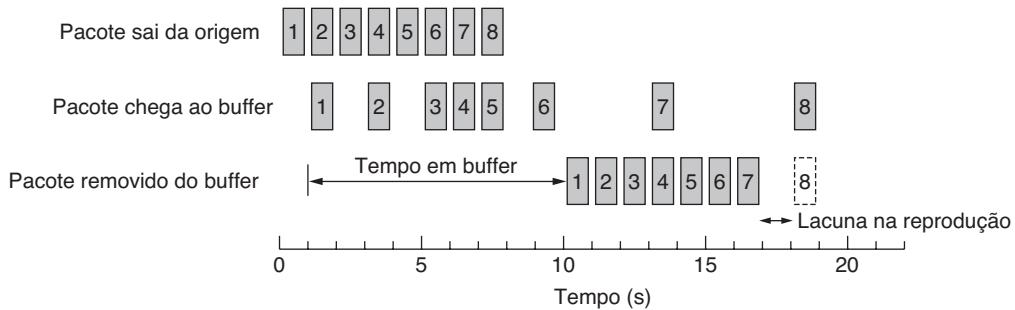



Figura 6.28 | Suavizando o fluxo de saída, mantendo pacotes em buffer.

do uma parada incômoda na música ou no filme. Em uma aplicação de mídia ao vivo, como em uma chamada VoIP, o pacote normalmente será pulado. Aplicações ao vivo não funcionam bem se forem interrompidas. Em uma aplicação de streaming de mídia, o player pode interromper. Esse problema pode ser aliviado atrasando o tempo ainda mais, usando um buffer maior. Para um player com streaming de áudio ou vídeo, os buffers de cerca de 10 segundos normalmente são usados para garantir que o player receba todos os pacotes (que não são descartados na rede) em tempo. Para aplicações ao vivo, como videoconferência, buffers curtos são necessários para garantir resposta em tempo.

Uma consideração importante para a reprodução suave é o **ponto de reprodução**, ou quanto tempo esperar pela mídia no receptor antes de iniciar a reprodução. Essa decisão depende do jitter. A diferença entre uma conexão com jitter baixo e jitter alto pode ser vista na Figura 6.29. O atraso médio pode não diferir muito dos dois, mas, se houver um jitter alto, o ponto de reprodução pode ser muito mais adiante, para capturar 99 por cento dos pacotes, do que se houvesse um jitter baixo.

Para escolher um bom ponto de reprodução, a aplicação pode medir o jitter examinando a diferença entre os períodos de tempo e a hora da chegada. Cada diferença oferece um exemplo de atraso (mais um deslocamento qualquer, fixo). No entanto, o atraso pode mudar com o tempo, devido a outras rotas de tráfego concorrentes e variáveis. Para acomodar essa mudança, as aplicações podem adaptar seu ponto de reprodução enquanto estão em execução. Porém, se isso não for benfeito, mudar o ponto de reprodução pode produzir um efeito observável ao usuário. Um modo de evitar o problema para o áudio é adaptar o ponto de reprodução entre os **períodos de fala**, nos intervalos de uma conversa. Ninguém notará a diferença entre um silêncio curto e outro ligeiramente maior. O RTP permite que as aplicações definam o bit marcador *M* para indicar o início de um novo período de fala com essa finalidade.

Se o atraso absoluto até que a mídia seja reproduzida for muito longo, as aplicações ao vivo sofrerão. Nada pode ser feito para reduzir o atraso de propagação se já estiver sendo usado um caminho direto. O ponto de reprodução pode ser atraído simplesmente aceitando-se que uma fração maior de pacotes chegará muito tarde para ser reproduzida. Se isso não for aceitável, a única maneira de atrair o ponto de reprodução é reduzir o jitter usando uma qualidade de serviço melhor, por exemplo, o serviço diferenciado de encaminhamento expresso. Ou seja, é necessário haver uma rede melhor.

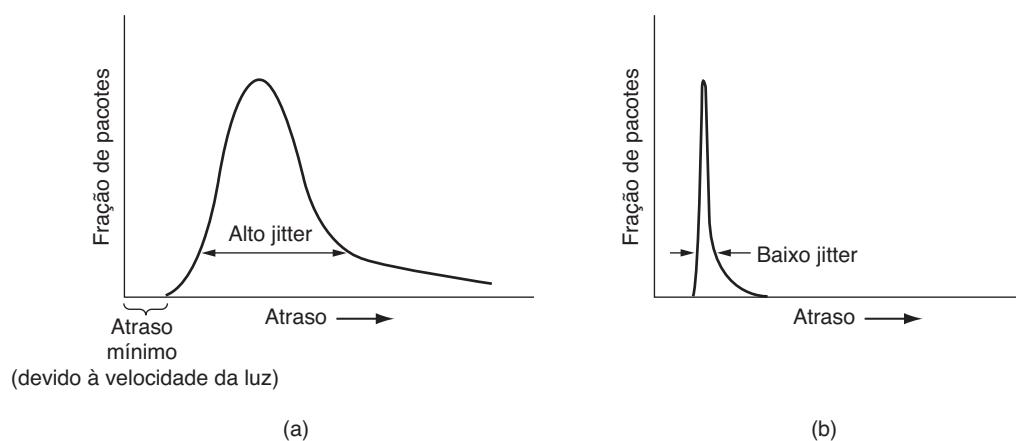



Figura 6.29 | (a) Alto jitter. (b) Baixo jitter.

6.5 Os protocolos de transporte da Internet: TCP

O UDP é um protocolo simples e tem alguns usos muito importantes, como interações cliente-servidor e multimídia; porém, para a maioria das aplicações da Internet, é necessária uma entrega confiável e em sequência. O UDP não pode proporcionar isso, e assim foi preciso criar outro protocolo. Ele se chama TCP e é o principal elemento da Internet. Vamos estudá-lo em detalhes nas próximas seções.

6.5.1 Introdução ao TCP

O protocolo de controle de transmissão, ou **TCP** (**Transmission Control Protocol**), foi projetado especificamente para oferecer um fluxo de bytes fim a fim confiável em uma rede interligada não confiável. Uma rede interligada é diferente de uma única rede porque suas diversas partes podem ter topologias, larguras de banda, atrasos, tamanhos de pacote e outros parâmetros completamente diferentes. O TCP foi projetado para se adaptar dinamicamente às propriedades da rede interligada e ser robusto diante dos muitos tipos de falhas que podem ocorrer.

O TCP foi definido formalmente na RFC 793, em setembro de 1981. Com o passar do tempo, diversas melhorias foram realizadas, e vários erros e inconsistências foram corrigidos. Para dar uma ideia da extensão do TCP, as RFCs importantes agora são a RFC 793 mais; esclarecimentos e soluções de alguns bugs na RFC 1122; extensões para alto desempenho na RFC 1323; confirmações seletivas na RFC 2018; controle de congestionamento na RFC 2581; modificação de propósito dos campos de cabeçalho para qualidade de serviço na RFC 2873; melhores sincronizações de retransmissão na RFC 2988; e notificação explícita de congestionamento na RFC 3168. A coleção completa é ainda maior, o que levou a um guia para as muitas RFCs, publicado naturalmente como outro documento RFC, a RFC 4614.

Cada máquina compatível com o TCP tem uma entidade de transporte TCP, que pode ser um procedimento de biblioteca, um processo do usuário ou parte do núcleo do sistema. Em todos os casos, ele gerencia fluxos e interfaces TCP para a camada IP. Uma entidade TCP aceita fluxos de dados do usuário provenientes de processos locais, divide-os em partes de no máximo 64 Kb (na prática, geralmente temos 1.460 bytes de dados, para que ele possa caber em um único quadro Ethernet com os cabeçalhos IP e TCP) e envia cada parte em um datagrama IP distinto. Quando os datagramas IP que contêm dados TCP chegam a uma máquina, eles são enviados à entidade TCP, que restaura os fluxos de bytes originais. Para simplificar, às vezes utilizamos apenas 'TCP', a fim de fazer referência tanto à entidade de transporte TCP (um software) quanto ao protocolo TCP (um conjunto de regras). Pelo contexto, ficará claro a qual deles estaremos nos referindo. Por exemplo, em 'O usuário envia os dados para TCP', está claro que estamos nos referindo à entidade de transporte TCP.

A camada IP não oferece nenhuma garantia de que os datagramas serão entregues da forma apropriada, nem indicação alguma da velocidade com que os datagramas podem ser enviados. Cabe ao TCP enviar datagramas com velocidade suficiente para utilizar a capacidade, mas sem causar congestionamento, além de definir o timeout aceito e retransmitir quaisquer datagramas que não serão entregues. Os datagramas também podem chegar fora de ordem; o TCP também terá de reorganizá-los em mensagens na sequência correta. Resumindo, o TCP deve fornecer bom desempenho com a confiabilidade que a maioria das aplicações deseja, mas que o IP não oferece.

6.5.2 O modelo de serviço do TCP

O serviço TCP é obtido quando tanto o transmissor como o receptor criam pontos extremos chamados **soquetes**, como vimos na Seção 6.1.3. Cada soquete tem um número (endereço) que consiste no endereço IP do host e em um número de 16 bits local para esse host, chamado **porta**. Porta é o nome usado pelo TCP para um TSAP. Para que o serviço TCP funcione, é necessário que uma conexão seja explicitamente estabelecida entre um soquete da máquina transmissora e um soquete da máquina receptora. As chamadas de soquetes estão listadas na Tabela 6.2.

Um soquete pode ser utilizado por várias conexões ao mesmo tempo. Em outras palavras, duas ou mais conexões podem terminar no mesmo soquete. As conexões são identificadas nas duas extremidades pelos identificadores de soquetes, ou seja, (*soquete1*, *soquete2*). Nenhum número de circuito virtual ou qualquer outro identificador é usado.

As portas com números abaixo de 1.024 são reservadas para serviços padronizados, que normalmente só podem ser iniciados por usuários privilegiados (por exemplo, root em sistemas UNIX). Elas são denominadas **portas conhecidas**. Por exemplo, qualquer processo que deseja recuperar remotamente o correio de um host pode se conectar à porta 143 do host de destino para entrar em contato com seu daemon IMAP. A lista de portas conhecidas é dada em *www.iana.org*. Mais de 700 já foram atribuídas. Algumas das mais conhecidas estão listadas na Tabela 6.4.

Outras portas de 1.024 a 49.151 podem ser registradas na IANA para ser usada por usuários privilegiados, mas as aplicações podem escolher e escolhem suas próprias portas. Por exemplo, a aplicação de compartilhamento de arquivos peer-to-peer BitTorrent (não oficialmente) usa as portas 6881-6887, mas também pode trabalhar com outras portas.

Certamente seria possível fazer o daemon FTP se associar à porta 21 durante a inicialização, fazer o daemon SSH se associar à porta 22 em tempo de inicialização e assim por diante. Porém, isso ocuparia a memória com daemons que ficariam ociosos na maior parte do tempo. Em vez disso, geralmente se tem um único, chamado **inetd** (**Internet daemon**) no UNIX, que se associa a várias portas e espera pela primeira conexão de entrada. Quando isso ocorre, o *inetd*

Porta	Protocolo	Uso
20, 21	FTP	Transferência de arquivos
22	SSH	Login remoto, substituto do Telnet
25	SMTP	Correio eletrônico
80	HTTP	World Wide Web
110	POP-3	Acesso remoto a correio eletrônico
143	IMAP	Acesso remoto a correio eletrônico
443	HTTPS	Web segura (HTTP sobre SSL/TLS)
543	RTSP	Controle de player de mídia
631	IPP	Compartilhamento de impressora

Tabela 6.4 | Algumas portas atribuídas.

ativa um novo processo e executa nele o daemon apropriado, deixando-o tratar a solicitação. Desse modo, os daemons diferentes de *inetd* só estão ativos quando há trabalho para eles realizarem. O *inetd* descobre que porta deve usar a partir de um arquivo de configuração. Consequentemente, o administrador do sistema pode configurar o sistema para ter daemons permanentes nas portas mais ocupadas (por exemplo, a porta 80) e *inetd* nas restantes.

Todas as conexões TCP são full-duplex e ponto a ponto. Full-duplex quer dizer que o tráfego pode ser feito em ambas as direções ao mesmo tempo. Ponto a ponto significa que cada conexão possui exatamente dois pontos terminais. O TCP não admite os processos de multicasting ou broadcasting.

Uma conexão TCP é um fluxo de bytes e não um fluxo de mensagens. As fronteiras das mensagens não são preservadas de uma extremidade a outra. Por exemplo, se o processo transmissor executar quatro gravações de 512 bytes em um fluxo TCP, esses dados poderão ser entregues ao processo receptor em quatro partes de 512 bytes, em duas de 1.024 bytes, uma de 2.048 bytes (ver Figura 6.30) ou em qualquer outra divisão. Não há um meio de o receptor detectar a(s) unidade(s) em que os dados foram gravados, não importa quanto ele tente.

No UNIX, os arquivos também têm essa propriedade. O leitor de um arquivo não é capaz de distinguir se ele foi gravado um bloco por vez, um byte por vez ou todo de uma vez. A exemplo do que acontece com um arquivo UNIX, o software TCP não tem ideia do significado dos bytes, e também não está interessado em descobri-lo. Um byte é apenas um byte.

Quando uma aplicação repassa dados para a entidade TCP, ela pode enviá-los imediatamente ou armazená-los em um buffer (para aguardar outros dados e enviar um volume maior de uma só vez), de acordo com suas necessidades. Entretanto, há ocasiões em que a aplicação realmente quer que os dados sejam enviados de imediato. Por exemplo, suponha que um usuário de um jogo interativo queira enviar um fluxo de atualizações. É essencial que as atualizações sejam enviadas imediatamente, e não mantidas em buffer até que haja uma coleção delas. Para forçar a saída dos dados, o TCP tem uma flag PUSH, que é transportado nos pacotes. A intenção original foi permitir que as aplicações digam às implementações TCP por meio do flag PUSH para não adiar a transmissão. Porém, as aplicações não podem literalmente definir a flag PUSH quando enviarem dados. Em vez disso, diferentes sistemas operacionais criaram diferentes opções para agilizar a transmissão (por exemplo, TCP_NODELAY em Windows e Linux).

Para os arqueólogos da Internet, também mencionaremos um recurso interessante do serviço TCP que permanece no protocolo, mas que raramente é usado: **dados urgentes**. Quando uma aplicação tem dados de alta prioridade, que devem ser processados imediatamente — por exemplo, se um usuário interativo pressionar a combinação de teclas CTRL-C para interromper uma computação remota que já foi iniciada —, a aplicação transmissora pode colocar alguma informação de controle no fluxo de dados e lhe passar para o TCP junto com a flag URGENT. Esse evento faz com que o TCP pare de acumular dados e transmita tudo que tem para essa conexão imediatamente.

Quando os dados urgentes são recebidos no destino, a aplicação receptora é interrompida (na terminologia UNIX, ela recebe um sinal) e para tudo o que estiver fazendo para ler o fluxo de dados e encontrar os dados urgentes. O final dos dados urgentes é marcado para que a aplicação saiba quando eles terminarem. O início dos dados urgentes não é marcado, e a aplicação deve saber identificá-lo.

Basicamente, esse esquema oferece um mecanismo de sinalização pouco sofisticado, deixando a maior parte do trabalho para a aplicação. Porém, embora os dados urgen-

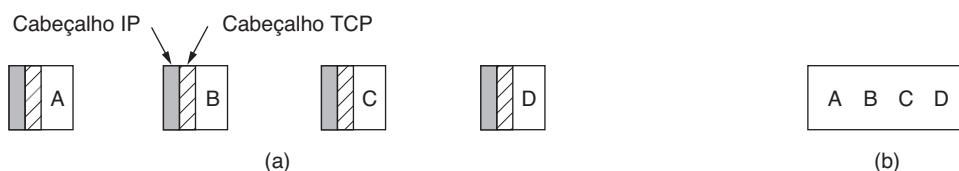



Figura 6.30 | (a) Quatro segmentos de 512 bytes enviados como datagramas IP separados. (b) Os 2.048 bytes de dados entregues à aplicação em uma única chamada READ.

tes sejam potencialmente úteis, eles não encontraram uma aplicação atraente e por isso caíram em desuso. Seu uso agora é desencorajado devido a diferenças de implementação, deixando que as aplicações lidem com sua própria sinalização. Talvez os protocolos de transporte futuros ofereçam melhor sinalização.

6.5.3 O protocolo TCP

Nesta seção, apresentaremos uma visão geral do protocolo TCP. Na próxima, veremos o cabeçalho do protocolo, campo a campo.

Uma característica fundamental do TCP, que domina o projeto do protocolo, é que cada byte em uma conexão TCP tem seu próprio número de sequência de 32 bits. Quando a Internet começou, as linhas entre roteadores eram principalmente linhas dedicadas de 56 kbps, e assim um host funcionando a toda a velocidade demorava mais de uma semana para percorrer todos os números de sequência. Na velocidade das redes modernas, os números de sequência podem ser consumidos a uma taxa alarmante, como veremos mais adiante. São usados números de sequência de 32 bits separados para a posição da janela deslizante em um sentido e para confirmações no sentido oposto, como descreveremos a seguir.

As entidades transmissoras e receptoras do TCP trocam dados na forma de segmentos. Um **segmento TCP** consiste em um cabeçalho fixo de 20 bytes (além de uma parte opcional), seguido por zero ou mais bytes de dados. O software TCP decide qual deve ser o tamanho dos segmentos. Ele pode acumular dados de várias gravações em um único segmento ou dividir os dados de uma única gravação em vários. Dois fatores restringem o tamanho do segmento. Primeiro, cada um, incluindo o cabeçalho do TCP, deve caber na carga útil do IP, que é de 65.515 bytes. Segundo, cada enlace tem uma unidade máxima de transferência, ou **MTU (Maximum Transfer Unit).** Cada segmento deve caber na MTU no transmissor e receptor, de modo que possa ser enviado e recebido em um único pacote, não fragmentado. Na prática, a MTU geralmente tem 1.500 bytes (o tamanho da carga útil Ethernet) e, portanto, define o limite superior de tamanho dos segmentos.

Porém, ainda é possível que os pacotes IP transportando segmentos TCP sejam fragmentados ao passar por um caminho da rede para o qual algum enlace tenha uma MTU pequena. Se isso acontecer, o desempenho é diminuído e causa outros problemas (Kent e Mogul, 1987). Em vez disso, as implementações TCP modernas realizam **descoberta da MTU do caminho** usando a técnica explicada na RFC 1191, que foi descrita na Seção 5.5.5. Essa técnica usa mensagens de erro ICMP para encontrar a menor MTU para qualquer enlace no caminho. O TCP então ajusta o tamanho do segmento para baixo, para evitar fragmentação.

O protocolo básico utilizado pelas entidades TCP é o de janela deslizante. Quando envia um segmento, o transmissor também dispara um timer. Quando o segmento chega ao destino, a entidade TCP receptora retorna um segmento (com ou sem dados, de acordo com as circunstâncias) com um número de confirmação igual ao próximo número de sequência que espera receber e o tamanho da janela restante. Se o timer do transmissor expirar antes de a confirmação ser recebida, o segmento será retransmitido.

Apesar de esse protocolo parecer simples, há muitos detalhes sobre ele que veremos a seguir. Os segmentos podem chegar fora de ordem; assim, os bytes 3.072 a 4.095 podem chegar, mas não podem ser confirmados, porque os bytes 2.048 a 3.071 ainda não chegaram. Além disso, os segmentos podem se atrasar tanto que o timer do transmissor expira e ele tem de retransmiti-los. As retransmissões podem incluir diferentes faixas de bytes em relação à transmissão original, exigindo uma administração cuidadosa para controlar quais bytes foram recebidos corretamente. Porém, como cada byte no fluxo tem seu próprio deslocamento exclusivo, isso pode ser feito.

O TCP deve estar preparado para lidar com todos esses problemas e resolvê-los de maneira eficiente. Foi feito um grande esforço no sentido de otimizar o desempenho dos fluxos TCP, mesmo diante dos problemas da rede. A seguir, descreveremos diversos algoritmos usados por muitas implementações do TCP.

6.5.4 O cabeçalho do segmento do TCP

A Figura 6.31 mostra o layout de um segmento TCP. Cada segmento começa com um cabeçalho de formato fixo de 20 bytes. O cabeçalho fixo pode ser seguido por opções de cabeçalho. Depois das opções, se for o caso, pode haver até 65.535 – 20 – 20 = 65.495 bytes de dados, onde o primeiro valor 20 se refere ao cabeçalho IP e o segundo ao cabeçalho TCP. Segmentos sem nenhum dado são válidos e são comumente usados para confirmações e mensagens de controle.

Vamos analisar o cabeçalho TCP campo a campo. Os campos *Porta de origem* e *Porta de destino* identificam os pontos terminais da conexão. A porta TCP mais o endereço IP de seu host formam uma extremidade exclusiva de 48 bits. As extremidades de origem e destino juntas identificam a conexão. Esse identificador de conexão é chamado **quíntuplas** pois consiste em cinco partes de informação: o protocolo (TCP), IP de origem e porta de origem, e IP de destino e porta de destino.

Os campos *Número de sequência* e *Número de confirmação* desempenham suas funções habituais. Observe que o segundo especifica o próximo byte esperado e não o último byte recebido corretamente. Ele é uma **confirmação acumulativa**, pois resume os dados recebidos com um único número. Ele não vai além dos dados perdidos. Ambos têm 32 bits, pois cada byte de dados é numerado em um fluxo TCP.

O campo *Comprimento do cabeçalho TCP* informa quantas palavras de 32 bits existem no cabeçalho TCP. Essa infor-

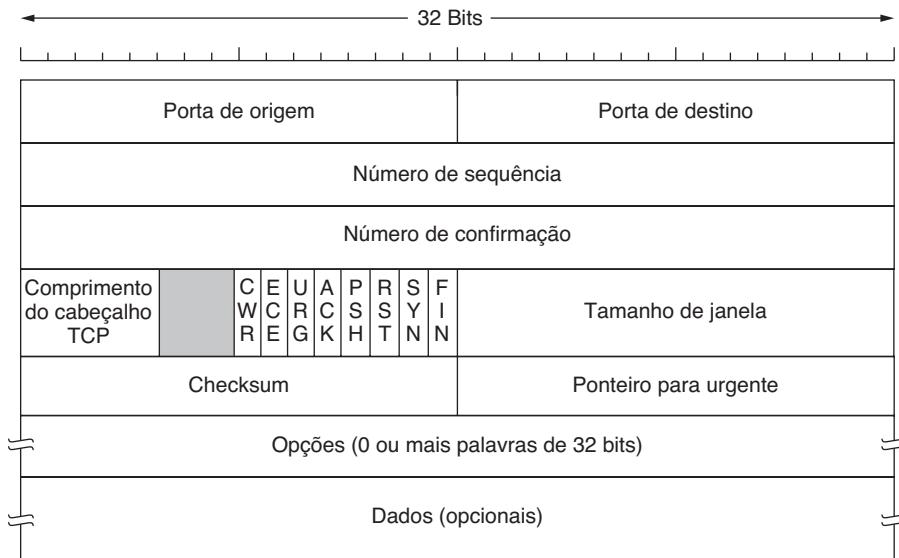



Figura 6.31 | O cabeçalho TCP.

mação é necessária, porque o campo *Opções* tem tamanho variável; assim, o mesmo acontece com o cabeçalho. Tecnicamente, na verdade, esse campo indica o início dos dados dentro do segmento com base em palavras de 32 bits, mas esse número é apenas o tamanho do cabeçalho em palavras e, portanto, o efeito é o mesmo.

Em seguida, temos um campo de 4 bits que não é utilizado. O fato de esse campo ter sobrevivido intacto por 30 anos (pois apenas 2 dos 6 bits reservados originais foram reivindicados) é a prova de como o TCP é bem organizado. Protocolos menores teriam precisado dele para corrigir bugs no projeto original.

Agora temos oito flags de 1 bit. *CWR* e *ECE* são usados para sinalizar congestionamento quando a ECN (Explicit Congestion Notification) é usada, conforme especificado na RFC 3168. A *ECE* é definida para sinalizar uma mensagem *ECN-Echo* a um transmissor TCP para solicitar a redução de velocidade quando o receptor TCP receber uma indicação de congestionamento da rede. A flag *CWR* é usada para sinalizar *Janela de congestionamento reduzida* do transmissor TCP para o receptor TCP, de modo que ele saiba que o transmissor diminuiu a velocidade e pode parar de enviar uma mensagem *ECN-Echo*. Discutiremos o papel da mensagem ECN no controle de congestionamento TCP na Seção 6.5.10.

O valor 1 é atribuído a *URG* se o *Ponteiro para urgente* estiver sendo usado. O *Ponteiro para urgente* é usado para indicar um deslocamento de bytes a partir do número de sequência atual em que os dados urgentes devem ser encontrados. Esse recurso substitui as mensagens de interrupção. Como já mencionamos, esse recurso representa uma forma estruturada de permitir que o transmissor envie um sinal ao receptor sem envolver o serviço TCP no motivo da interrupção, mas isso raramente é usado.

À flag *ACK* é atribuído o bit 1 para indicar que o *Número de confirmação* é válido. Isso acontece para quase todos os pacotes. Se *ACK* for igual a zero, isso significa que o segmento não contém uma confirmação e assim o campo *Número de confirmação* é ignorado.

A flag *PSH* indica dados com PUSH. Com ele, o receptor é solicitado a entregar os dados à aplicação mediante sua chegada, em vez de armazená-los até que um buffer completo tenha sido recebido (o que ele poderia fazer para manter a eficiência).

A flag *RST* é utilizada para reiniciar uma conexão que tenha ficado confusa devido a uma falha no host ou por qualquer outra razão. A RST também é utilizada para rejeitar um segmento inválido ou para recusar uma tentativa de conexão. Em geral, se receber um segmento com o bit *RST* ativado, isso significa que você tem um problema.

A flag *SYN* é usada para estabelecer conexões. A solicitação de conexão tem *SYN* = 1 e *ACK* = 0 para indicar que o campo de confirmação piggyback não está sendo utilizado. A resposta contém uma confirmação e, portanto, tem *SYN* = 1 e ACK = 1. Basicamente, o bit *SYN* é usado para indicar CONNECTION REQUEST e CONNECTION ACCEPTED, enquanto o bit *ACK* é usado para distinguir entre essas duas possibilidades.

A flag *FIN* é utilizada para encerrar uma conexão. A flag *FIN* indica que o transmissor não tem mais dados para *transmitir*. Entretanto, um processo pode continuar a receber dados indefinidamente, mesmo depois que a conexão tiver sido encerrada. Tanto o segmento *SYN* quanto o segmento *FIN* têm números de sequência e, portanto, são processados na ordem correta.

O controle de fluxo no TCP é administrado por meio de uma janela deslizante de tamanho variável. O campo

Tamanho de janela indica quantos bytes podem ser enviados a partir do byte confirmado. Um campo *Tamanho de janela* igual a 0 é valido e informa que todos os bytes até *Número de confirmação* – 1, inclusive, foram recebidos, mas que o receptor precisa de um descanso no momento e agradeceria muito se nenhum outro dado fosse enviado. O receptor pode mais tarde conceder permissão para enviar, transmitindo um segmento com o mesmo *Número de confirmação* e com um campo *Tamanho de janela* diferente de zero.

Nos protocolos do Capítulo 3, as confirmações de quadros recebidos e a permissão para enviar novos quadros eram mantidas juntas. Isso era uma consequência do tamanho fixo da janela para cada protocolo. No TCP, as confirmações e a permissão para enviar dados adicionais são completamente isoladas. Na verdade, um receptor pode dizer: 'Recebi os bytes até *k*, mas não quero mais agora'. Esse desacoplamento (na verdade, uma janela de tamanho variável) proporciona flexibilidade adicional. Vamos estudá-lo em detalhes a seguir.

Um *Checksum* também é fornecido para aumentar a confiabilidade. Ele confere o checksum do cabeçalho, dos dados e do pseudocabeçalho exatamente da mesma maneira que o UDP, exceto que o pseudocabeçalho tem o número de protocolo para TCP (6) e o checksum é obrigatório. Para obter mais detalhes, consulte a Seção 6.4.1.

O campo *Opções* foi projetado como uma forma de oferecer recursos extras, ou seja, recursos que não foram previstos pelo cabeçalho comum. Muitas opções foram definidas e várias são comumente utilizadas. As opções são de tamanho variável, preenchem um múltiplo de 32 bits usando o preenchimento com zeros e podem se estender para 40 bytes para acomodar o maior cabeçalho TCP que pode ser especificado. Algumas opções são transportadas quando uma conexão é estabelecida para negociar ou informar o outro lado das capacidades. Outras opções são transportadas sobre pacotes durante o tempo de vida da conexão. Cada opção tem uma codificação de Tipo-Tamanho-Valor.

Uma opção largamente usada é aquela que permite a cada host estipular o tamanho máximo do segmento, ou **MSS** (**Maximum Segment Size**), que está disposto a receber. O uso de segmentos grandes é mais eficiente do que a utilização de segmentos pequenos, pois o cabeçalho de 20 bytes pode ser diluído em um maior volume de dados; porém, é possível que hosts pequenos não sejam capazes de administrar segmentos muito grandes. Durante a configuração da conexão, cada lado pode anunciar sua capacidade máxima e avaliar a capacidade de seu parceiro. Se um host não usar essa opção, o valor-padrão de 536 bytes será estipulado para a carga útil. Todos os hosts da Internet são obrigados a aceitar segmentos TCP de 536 + 20 = 556 bytes. O tamanho máximo do segmento nos dois sentidos não precisa ser o mesmo.

Para enlaces com alta largura de banda, alto atraso ou ambos, a janela de 64 KB, correspondente a um campo de 16 bits, é quase sempre um problema. Por exemplo, em uma linha OC-12 (de aproximadamente 600 Mbps), é necessário menos de 1 ms para enviar uma janela de 64 KB completa. Se o atraso de propagação da viagem de ida e volta for de 50 ms (o mais comum em um cabo de fibra óptica transcontinental), o transmissor ficará inativo durante mais de 98 por cento do tempo, aguardando confirmações. Um tamanho de janela maior permitiria que o transmissor continuasse enviando os dados. A opção **window scale** permite ao transmissor e ao receptor negociar um fator de escala no início de uma conexão. Os dois lados usam o fator de escala para deslocar o campo *Tamanho de janela* para 14 bits à esquerda, permitindo assim janelas de até 2^{30} bytes. A maior parte das implementações do TCP já é compatível com essa opção.

A opção **timestamp** (registro de tempo) transporta um período de tempo enviado pelo transmissor e ecoado pelo receptor. Ele está incluído em cada pacote, uma vez que seu uso é definido durante o estabelecimento da conexão, e usado para calcular as amostras de tempo de ida e volta que são usadas para estimar quando um pacote foi perdido. Ele também é usado como uma extensão lógica do número de sequência de 32 bits. Em uma conexão rápida, o número de sequência pode se esgotar rapidamente, ocasionando uma possível confusão entre dados novos e antigos. O esquema **PAWS** (**Protection Against Wrapped Sequence numbers**) descarta os segmentos que chegam com períodos de tempo antigos para evitar esse problema.

Finalmente, a opção **SACK** (**Selective ACKnowledgement**) permite que um receptor informe a um transmissor os intervalos de números de sequência que ele recebeu. Ele incrementa o *Número de confirmação* e é usado após um pacote ter sido perdido, mas após os dados subsequentes (ou duplicados) terem chegado. Os novos dados não aparecem no campo *Número de confirmação* no cabeçalho, pois esse campo oferece apenas o próximo byte esperado em ordem. Com o SACK, o transmissor está explicitamente ciente de quais dados o receptor tem e, portanto, pode determinar quais dados devem ser retransmitidos. O SACK é definido na RFC 2108 e na RFC 2883 e é usado cada vez mais. Descrevemos o uso do SACK junto com controle de congestionamento na Seção 6.5.10.

6.5.5 Estabelecimento de conexões TCP

As conexões são estabelecidas no TCP por meio do handshake de três vias discutido na Seção 6.2.2. Para estabelecer uma conexão, um lado — digamos, o servidor — aguarda passivamente por uma conexão de entrada, executando as primitivas LISTEN e ACCEPT, nessa ordem, através da especificação de determinada origem ou de ninguém em particular.

O outro lado — digamos, o cliente — executa uma primitiva CONNECT, especificando o endereço IP e a porta à

qual deseja se conectar, o tamanho máximo do segmento TCP que está disposto a aceitar e, opcionalmente, alguns dados do usuário (por exemplo, uma senha). A primitiva CONNECT envia um segmento TCP com o bit da flag *SYN* ativado e um bit da flag *ACK* desativado, e aguarda uma resposta.

Quando esse segmento chega ao destino, a entidade TCP dessa estação verifica se existe um processo que tenha executado uma primitiva LISTEN na porta informada no campo *Porta de destino*. Caso contrário, ela envia uma resposta com o bit da flag *RST* ativado para rejeitar a conexão.

Se algum processo estiver na escuta da porta, esse processo receberá o segmento TCP de entrada. Em seguida, ele poderá aceitar ou rejeitar a conexão. Se o processo aceitar, um segmento de confirmação será retornado. A sequência dos segmentos TCP enviados em condições normais é ilustrada na Figura 6.32(a). Observe que um segmento *SYN* consome 1 byte de espaço de sequência, para que seja confirmado sem ambiguidade.

No caso de dois hosts tentarem estabelecer uma conexão entre os mesmos soquetes simultaneamente, ocorrerá a sequência de eventos ilustrada na Figura 6.32(b). O resultado desses eventos é o estabelecimento de apenas uma conexão e não de duas, porque as conexões são identificadas por suas extremidades. Se a primeira configuração resultar em uma conexão identificada por (x, y) e a segunda também, haverá somente uma entrada na tabela, a saber, para (x, y).

Lembre-se de que o número de sequência inicial escolhido por cada host deve ser reciclado lentamente, em vez de ser uma constante como 0. Essa regra serve para proteger contra pacotes duplicados atrasados, conforme discutimos na Seção 6.2.2. Originalmente, isso era feito com um esquema baseado em clock, em que um pulso de clock ocorria a cada 4 μs.

Entretanto, uma vulnerabilidade com a implementação do handshake de três vias é que o processo que escuta precisa se lembrar de seu número de sequência assim que responde com seu próprio segmento *SYN*. Isso significa que um transmissor malicioso pode amarrar recursos em um host enviando um fluxo de segmentos *SYN* e nunca continuando para completar a conexão. Esse ataque é chamado **inundação de SYN**, e prejudicou muitos servidores Web na década de 90.

Um modo de proteger contra esse ataque é usar **cookies SYN**. Em vez de lembrar o número de sequência, um host escolhe um número de sequência gerado criptograficamente, coloca o no segmento de saída e se esquece dele. Se o handshake de três vias for concluído, esse número de sequência (mais 1) será retornado para o host. Ele pode então recriar o número de sequência correto executando a mesma função criptográfica, desde que as entradas para essa função sejam conhecidas, por exemplo, o endereço IP e a porta do outro host, e um segredo local. Esse procedimento permite que o host verifique se um número de sequência confirmado está correto sem ter que se lembrar do número de sequência separadamente. Existem alguns problemas, como a incapacidade de lidar com opções do TCP, de modo que os cookies SYN podem ser usados somente quando o host está sujeito a uma inundação de SYN. Porém, eles são uma guinada interessante no estabelecimento da conexão. Para obter mais informações, consulte a RFC 4987 e Lemon (2002).

6.5.6 Encerramento da conexão TCP

Apesar de as conexões TCP serem full-duplex, fica mais fácil compreender como as conexões são encerradas se as considerarmos um par de conexões simplex. Cada conexão simplex é encerrada de modo independente de sua parceira. Para encerrar uma conexão, qualquer um dos lados pode enviar um segmento com o bit *FIN* ativa-

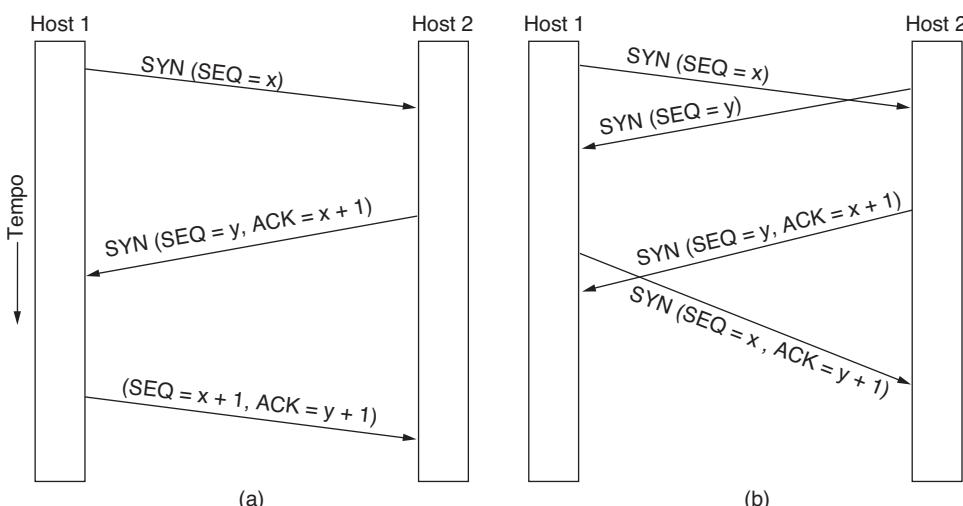



Figura 6.32 | (a) Estabelecimento da conexão TCP no caso normal. (b) Estabelecimento de conexão simultâneo nos dois lados.

do, o que significa que não há mais dados a ser transmitidos. Quando *FIN* é confirmado, esse sentido é desativado para novos dados. No entanto, os dados podem continuar a fluir indefinidamente no outro sentido. Quando os dois sentidos da conexão estiverem desativados, a conexão será encerrada. De modo geral, são necessários quatro segmentos TCP para encerrar uma conexão, isto é, um *FIN* e um *ACK* para cada sentido. Porém, é possível que o primeiro *ACK* e o segundo *FIN* ocupem o mesmo segmento, o que baixa o número total para três.

A exemplo do que ocorre com as ligações telefônicas em que duas pessoas se despedem e desligam simultaneamente, é possível que as duas extremidades da conexão TCP enviem segmentos *FIN* ao mesmo tempo. Eles são confirmados do modo habitual, e a conexão é desativada. Na verdade, não há nenhuma diferença essencial entre as situações em que os dois hosts encerram a conexão de forma sequencial ou simultânea.

Para evitar que ocorra o problema dos dois exércitos (discutido na Seção 6.2.3), são utilizados timers. Se uma resposta para um *FIN* não chegar no período equivalente a duas vezes o tempo máximo de duração de um pacote, o transmissor do *FIN* encerrará a conexão. Por fim, o outro lado perceberá que não há mais ninguém na escuta e também sofrerá um timeout. Mesmo que essa solução não seja perfeita, pois a solução perfeita é teoricamente impossível, ela terá de bastar. Na prática, os problemas são raros.

6.5.7 Modelagem e gerenciamento de conexões TCP

As etapas necessárias para o estabelecimento e o encerramento de conexões podem ser representadas em uma máquina de estados finitos com os 11 estados mostrados na Tabela 6.5. Em cada estado, determinados eventos são válidos. Quando ocorre um evento válido, é possível executar uma ação. Se ocorrer algum outro evento, será reportado um erro.

Cada conexão começa no estado *CLOSED*. Ela sai desse estado ao executar uma abertura passiva (LISTEN) ou ativa (CONNECT). Se o outro lado executar a primitiva oposta, a conexão será estabelecida e o estado passará a ser *ESTABLISHED*. O encerramento da conexão pode ser iniciado por qualquer um dos lados. Quando é completado, o estado volta a ser *CLOSED*.

A máquina de estados finitos propriamente dita está ilustrada na Figura 6.33. Uma situação comum em que um cliente se conecta ativamente a um servidor passivo é representada pelas linhas mais escuras — a linha contínua para o cliente e a linha tracejada para o servidor. As linhas mais claras representam sequências de eventos incomuns. Cada linha na Figura 6.33 é marcada por um par *evento/ação*. O evento pode ser uma chamada de sistema iniciada pelo usuário (CONNECT, LISTEN, SEND ou CLOSE), uma chegada de segmento (*SYN*, *FIN*, *ACK* ou *RST*) ou, em um caso único, um período de timeout igual a duas vezes a duração máxima dos pacotes. A ação é o envio de um segmento de controle (*SYN*, *FIN* ou *RST*) ou nada, indicado por um travessão (—). Os comentários são mostrados entre parênteses.

O diagrama pode ser mais bem compreendido se seguirmos primeiro o caminho de um cliente (a linha escura contínua) e depois o caminho do servidor (a linha escura tracejada). Quando um programa de aplicação na máquina cliente emite uma solicitação CONNECT, a entidade TCP local cria um registro de conexão, assinala que a conexão se encontra no estado *SYN SENT* e envia um segmento *SYN*. Observe que muitas conexões podem es-

Estado	Descrição
CLOSED	Nenhuma conexão ativa ou pendente
LISTEN	O servidor está esperando a chegada de uma chamada
SYN RCVD	Uma solicitação de conexão chegou; espera por ACK
SYN SENT	A aplicação começou a abrir uma conexão
ESTABLISHED	O estado normal para a transferência de dados
FIN WAIT 1	A aplicação informou que terminou de transmitir
FIN WAIT 2	O outro lado concordou em encerrar
TIME WAIT	Aguarda a entrega de todos os pacotes
CLOSING	Ambos os lados tentaram encerrar a transmissão simultaneamente
CLOSE WAIT	O outro lado deu início a um encerramento
LAST ACK	Aguarda a entrega de todos os pacotes

Tabela 6.5 | Os estados usados na máquina de estados finitos para o gerenciamento de uma conexão TCP.

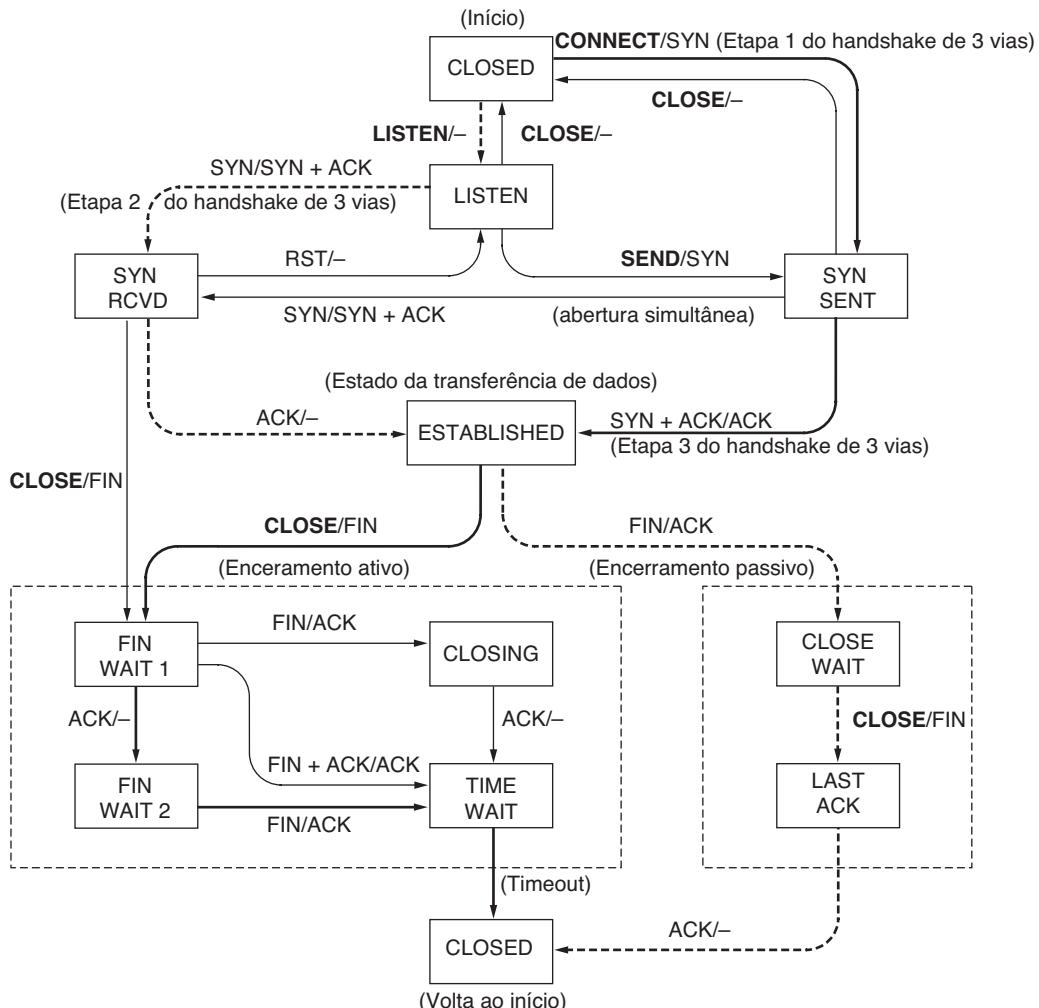



Figura 6.33 | Máquina de estados finitos usada no gerenciamento de uma conexão TCP. A linha contínua mais escura representa o caminho normal de um cliente. A linha tracejada mais escura representa o caminho normal de um servidor. As linhas mais finas representam eventos incomuns. Cada transição é identificada pelo evento que a provoca e pela ação resultante dela, separados por uma barra.

tar abertas (ou sendo abertas) ao mesmo tempo por várias aplicações; portanto, o estado se refere a cada conexão e é incluído no registro de conexões. Quando *SYN + ACK* chega, o TCP envia o *ACK* final do handshake de três vias e passa para o estado *ESTABLISHED*. Só então os dados podem ser transmitidos e recebidos.

Quando uma aplicação é encerrada, ela executa uma primitiva CLOSE, o que faz com que a entidade TCP local envie um segmento *FIN* e aguarde o *ACK* correspondente (o quadro tracejado marca o 'encerramento ativo'). Quando *ACK* chega, há uma transição para o estado *FIN WAIT 2* e um sentido da conexão é desativado. Quando o outro lado também for desativado, chegará um *FIN* que será confirmado. Agora os dois lados estão desativados, mas o TCP aguarda um período equivalente ao tempo máximo de duração de um pacote para ter certeza de que todos os pacotes da conexão foram recebidos, caso alguma confirmação tenha se perdido. Quando o timer expirar, o TCP removerá o registro da conexão.

Examinaremos agora o gerenciamento de conexões do ponto de vista do servidor. O servidor executa uma primitiva LISTEN e aguarda para ver quem aparece. Quando um *SYN* chegar, ele será confirmado e o servidor passará para o estado *SYN RCVD*. Quando o *SYN* do servidor for confirmado, o handshake de três vias estará completo e o servidor passará para o estado *ESTABLISHED*. Nesse caso, a transferência de dados já pode ocorrer.

Quando acabar de transmitir seus dados, o cliente executará uma primitiva CLOSE, o que faz com que um *FIN* chegue ao servidor (o quadro tracejado marca o 'encerramento passivo'). Em seguida, o servidor recebe um sinal. Quando ele também executar uma primitiva CLOSE, um *FIN* será enviado ao cliente. Quando a confirmação do cliente for recebida, o servidor encerrará a conexão e apagará o registro da conexão.

6.5.8 Janela deslizante do TCP

Como mencionamos antes, o gerenciamento de janelas no TCP desvincula as questões de confirmação do recebimento correto dos segmentos da alocação de buffer pelo receptor. Por exemplo, suponha que o receptor tenha um buffer de 4.096 bytes, como ilustra a Figura 6.34. Se o transmissor enviar um segmento de 2.048 bytes e este for recebido de forma correta, o receptor confirmará o segmento. Porém, como agora ele só tem 2.048 bytes de espaço disponível em seu buffer (até que alguma aplicação retire alguns dados do buffer), o receptor anunciará uma janela de 2.048 bytes começando no próximo byte esperado.

Agora, o transmissor envia outros 2.048 bytes, que são confirmados, mas a janela anunciada tem tamanho 0. O transmissor deve parar até que o processo da aplicação no host receptor tenha removido alguns dados do buffer, quando o TCP poderá anunciar uma janela maior e mais dados poderão ser enviados.

Quando a janela é 0, o transmissor não pode enviar segmentos da forma como faria sob condições normais, mas há duas exceções. Primeira, os dados urgentes podem ser enviados para, por exemplo, permitir que o usuário encerre o processo executado na máquina remota. Segunda, o transmissor pode enviar um segmento de 1 byte para fazer com que o receptor anuncie novamente o próximo byte esperado e o tamanho da janela. Esse pacote é chamado **window probe**. O padrão TCP oferece essa opção de forma explícita para evitar um impasse no caso de uma atualização da janela se perder.

Os transmissores não são obrigados a enviar os dados assim que os recebem da aplicação. Nem os receptores têm a obrigação de enviar as confirmações imediatamente. Por exemplo, na Figura 6.34, quando os primeiros 2 KB de dados chegaram, o TCP, sabendo que havia uma janela de 4 KB disponível, estaria completamente correto se apenas armazenasse os dados no buffer até chegarem outros 2 KB, a fim de poder transmitir um segmento com 4 KB de carga útil. Essa liberdade pode ser explorada para melhorar o desempenho das conexões.

Considere uma conexão para um terminal remoto, por exemplo, usando SSH ou Telnet, que reage a cada tecla pressionada. Na pior das hipóteses, quando um caractere chegar à entidade TCP receptora, o TCP cria um segmento TCP de 21 bytes, que será repassado ao IP para ser enviado como um datagrama IP de 41 bytes. No lado receptor, o TCP envia imediatamente uma confirmação de 40 bytes (20 bytes de cabeçalho TCP e 20 bytes de cabeçalho IP). Mais tarde, quando o terminal remoto tiver lido o byte, o

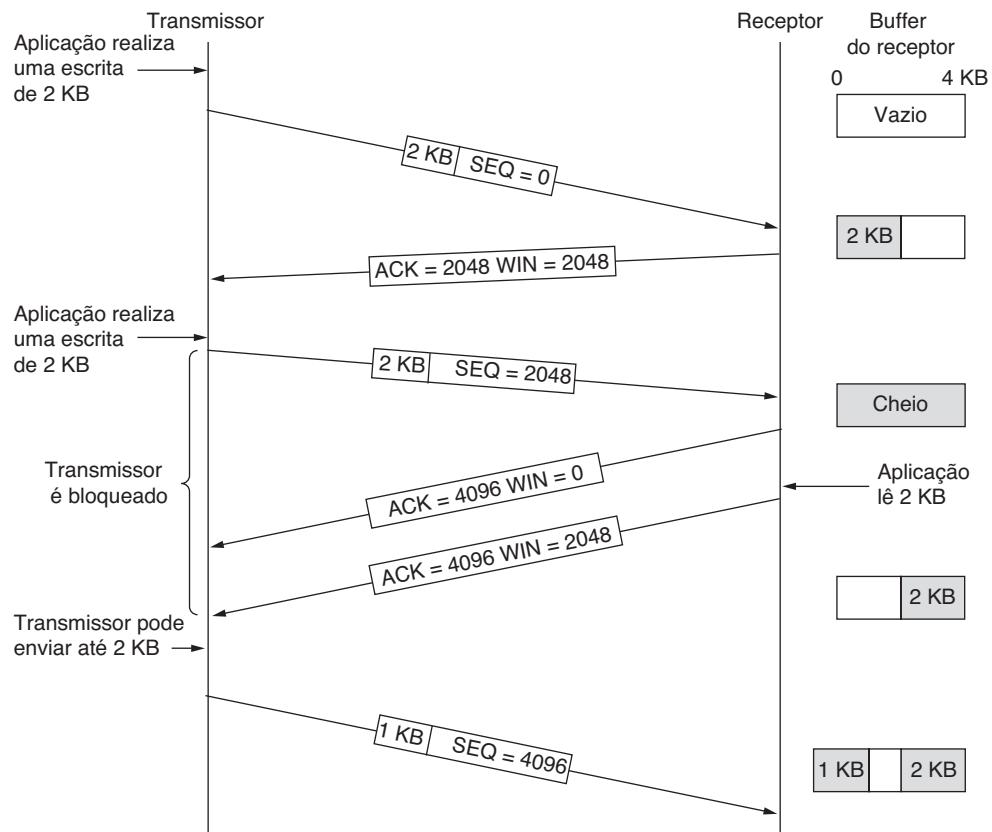



Figura 6.34 | Gerenciamento de janelas no TCP.

TCP enviará uma atualização da janela, movendo a janela um byte para a direita. Esse pacote também tem 40 bytes. Por último, quando o terminal tiver processado o caractere, ele o ecoará para exibição local usando um pacote de 41 bytes. No total, 162 bytes de largura de banda são utilizados e quatro segmentos são enviados para cada caractere digitado. Quando a largura de banda é escassa, esse método não se mostra uma boa opção.

Uma abordagem usada por muitas implementações do TCP para otimizar essa situação é chamada **confirmações adiadas.** A ideia é retardar as confirmações e atualizações de janelas durante 500 ms, na esperança de encontrar algum dado que lhes dê 'uma carona'. Supondo que o terminal ecoe dentro de 500 ms, apenas um pacote de 41 bytes precisará ser retornado ao usuário remoto, reduzindo à metade a contagem de pacotes e o uso da largura de banda.

Embora essa regra reduza a carga imposta à rede pelo receptor, o transmissor ainda estará operando de modo ineficiente, enviando pacotes de 41 bytes que contêm apenas um byte de dados. Uma forma de reduzir esse uso é conhecida como **algoritmo de Nagle** (Nagle, 1984). A sugestão de Nagle é simples: quando os dados chegarem ao transmissor em pequenas partes, basta enviar o primeiro byte e armazenar no buffer todos os outros, até que o byte pendente tenha sido confirmado. Em seguida, envie todos os caracteres armazenados no buffer em um único segmento TCP e comece a armazenar no buffer novamente, até que o próximo segmento seja confirmado. Ou seja, somente um pacote pequeno pode estar pendente a qualquer momento. Se muitas partes dos dados forem enviadas pela aplicação em um tempo de ida e volta, o algoritmo de Nagle colocará as muitas partes em um segmento, reduzindo bastante a largura de banda utilizada. O algoritmo diz ainda que um novo segmento deveria ser enviado se dados bastantes completassem um segmento máximo.

O algoritmo de Nagle é muito utilizado por implementações TCP, mas há ocasiões em que é melhor desativá-lo. Em particular, em jogos interativos executados na Internet, os jogadores normalmente desejam um fluxo rápido de pequenos pacotes de atualização. Reunir as atualizações para enviá-las em rajadas faz com que o jogo responda indevidamente, o que deixa os usuários insatisfeitos. Um problema mais sutil é que o algoritmo de Nagle às vezes pode interagir com confirmações adiadas e causar um impasse temporário: o receptor espera por dados nos quais pode enviar uma confirmação de carona, e o transmissor espera a confirmação para enviar mais dados. Essa interação pode adiar os downloads de páginas Web. Devido a esses problemas, o algoritmo de Nagle pode ser desativado (uma opção chamada *TCP_NODELAY*). Mogul e Minshall (2001) discutem essa e outras soluções.

Outro problema que pode arruinar o desempenho do TCP é a **síndrome do janelamento inútil** (Clark, 1982). Esse problema ocorre quando os dados são passados para a entidade TCP transmissora em grandes blocos, mas uma aplicação interativa no lado receptor lê os dados apenas um byte por vez. Para entender o problema, observe a Figura 6.35. Inicialmente, o buffer TCP no lado receptor está cheio (ou seja, ele tem uma janela de tamanho 0) e o transmissor sabe disso. Em seguida, uma aplicação interativa lê um caractere do fluxo TCP. Essa ação faz com que o TCP receptor fique satisfeito e envie uma atualização da janela ao transmissor, informando que ele pode enviar 1 byte. O transmissor agradece e envia 1 byte. Agora, o buffer se enche outra vez; portanto, o receptor confirma o segmento de 1 byte e atribui o valor 0 ao tamanho da janela. Esse comportamento pode durar para sempre.

A solução apresentada por Clark é evitar que o receptor envie uma atualização da janela de 1 byte. Em vez disso, ele é forçado a aguardar até que haja um espaço considerável na janela para então anunciar o fato. Para ser mais específico, o receptor não deve enviar uma atualização da janela até que possa lidar com o tamanho máximo do segmento que anunciou quando a conexão foi estabelecida, ou até que seu buffer esteja com metade de sua capacidade livre; o que for menor. Além disso, o transmissor também pode ajudar não enviando segmentos muito pequenos. Em vez disso, deve tentar aguardar até que possa enviar um segmento inteiro ou pelo menos um que contenha dados equivalentes à metade da capacidade do buffer no lado receptor.

O algoritmo de Nagle e a solução de Clark para a síndrome do janelamento inútil são complementares. Nagle tentava resolver o problema causado pelo fato de a aplicação transmissora entregar dados ao TCP um byte por vez. Já Clark tentava acabar com o problema criado pelo fato de a aplicação receptora retirar os dados do TCP um byte por vez. Ambas as soluções são válidas e podem funcionar juntas. O objetivo é evitar que o transmissor envie segmentos pequenos e que o receptor tenha de solicitá-los.

O TCP receptor pode fazer mais para melhorar o desempenho da rede do que apenas realizar a atualização das janelas em unidades grandes. Assim como o TCP transmissor, ele também tem a capacidade de armazenar dados em buffer; portanto, pode bloquear uma solicitação READ da aplicação até ter um bom volume de dados a oferecer. Isso reduz o número de chamadas ao TCP (e também o overhead). Isso também aumenta o tempo de resposta; no entanto, para aplicações não interativas, como a transferência de arquivos, a eficiência pode ser mais importante que o tempo de resposta a solicitações individuais.

Outro problema para o receptor é o que fazer com os segmentos fora de ordem. O receptor armazenará os dados em buffer até que possam ser passados para a aplicação em ordem. Na realidade, nada de mau aconteceria se os segmentos fora de ordem fossem descartados, pois eles por fim seriam retransmitidos pelo transmissor, mas isso seria um desperdício.

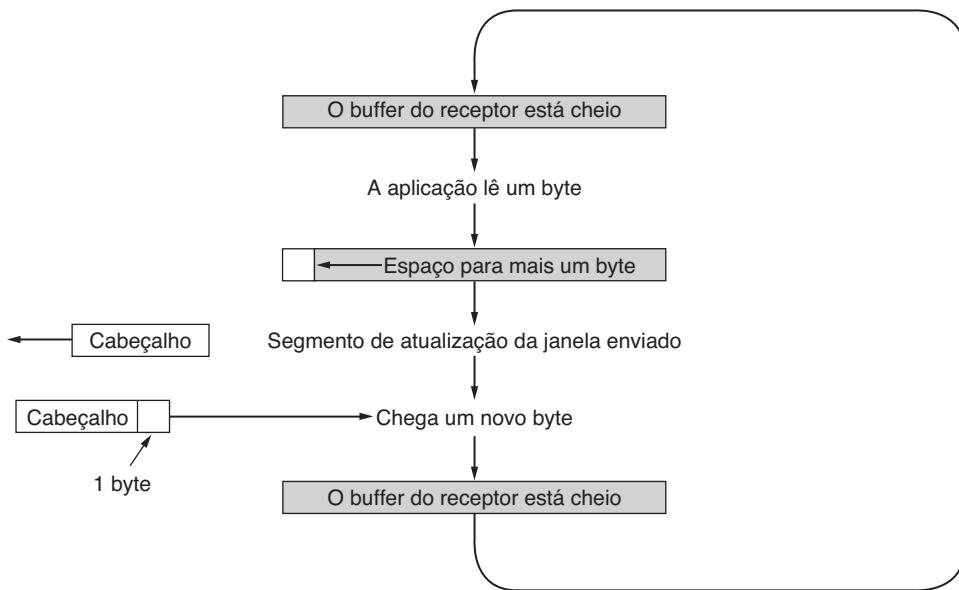



Figura 6.35 | Síndrome do janelamento inútil.

As confirmações só podem ser enviadas quando todos os dados até o byte confirmado tiverem sido recebidos. Isso é chamado **confirmação acumulativa**. Se o receptor receber os segmentos 0, 1, 2, 4, 5, 6 e 7, ele poderá confirmar tudo até o último byte do segmento 2, inclusive. Quando o transmissor sofrer um timeout, ele retransmitirá o segmento 3. Se tiver armazenado os segmentos de 4 a 7, o receptor poderá, ao receber o segmento 3, confirmar todos os bytes até o fim do segmento 7.

6.5.9 Gerenciamento de contadores do TCP

O TCP utiliza vários timers (pelo menos conceitualmente) para realizar seu trabalho. O mais importante deles é o timer de retransmissão, ou **RTO** (**Retransmission Timeout**). Quando um segmento é enviado, um timer de retransmissão é iniciado. Se o segmento for confirmado antes de o timer expirar, ele será interrompido. Por outro lado, se o timer expirar antes de a confirmação chegar, o segmento será retransmitido (e o timer disparado mais uma vez). Com isso, surge a seguinte pergunta: Qual deve ser esse período de tempo?

Esse problema é muito mais difícil na camada de transporte do que nos protocolos de enlace de dados, como o 802.11. Nesse último caso, o atraso esperado é medido em microssegundos e é altamente previsível (ou seja, tem pouca variância), de modo que o timer pode ser definido para expirar pouco depois da confirmação esperada, como mostra a Figura 6.36(a). Como as confirmações raramente são adiadas na camada de enlace de dados (devido à falta de congestionamento), a ausência de uma confirmação no momento esperado geralmente significa que ou o quadro ou a confirmação foram perdidos.

O TCP encontra um ambiente radicalmente distinto. A função densidade de probabilidade para o tempo gasto para uma confirmação TCP voltar é mais semelhante à Figura 6.36(b) do que à Figura 6.36(a). Ela é maior e mais variável. Determinar o tempo de ida e volta ao destino é complicado. Mesmo quando é conhecido, decidir sobre o intervalo de tempo também é difícil. Se o período de tempo for muito curto, digamos T_1 na Figura 6.36(b), haverá retransmissões desnecessárias, enchendo a Internet com pacotes inúteis. Se ele for muito longo (por exemplo, T_2), o desempenho sofrerá devido ao longo atraso de retransmissão sempre que o pacote se perder. Além do mais, a média e a variância da distribuição de chegada da confirmação podem mudar rapidamente dentro de alguns segundos, enquanto o congestionamento se acumula ou é resolvido.

A solução é usar um algoritmo dinâmico que adapte constantemente o intervalo de tempo, com base em medições contínuas do desempenho da rede. O algoritmo geralmente usado pelo TCP é atribuído a Jacobson (1988) e funciona da seguinte forma. Para cada conexão, o TCP mantém uma variável, *SRTT* (Smoothed Round-Trip Time), que é a melhor estimativa atual do tempo de ida e volta até o destino em questão. Quando um segmento é enviado, um timer é iniciado, tanto para ver quanto tempo a confirmação leva como também para disparar uma retransmissão se o tempo for muito longo. Se a confirmação retornar antes que o timer expire, o TCP mede o tempo gasto para

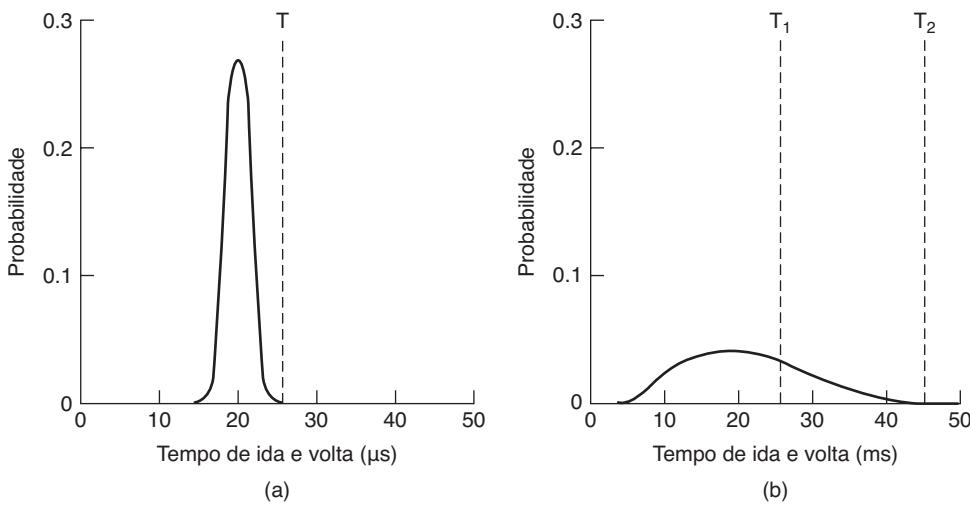



Figura 6.36 | (a) Densidade de probabilidade de tempos de chegada de confirmações na camada de enlace de dados. (b) Densidade de probabilidade de tempos de chegada de confirmações para o TCP.

a confirmação, digamos, *R*. Depois, ele atualiza o *SRTT* de acordo com a fórmula

$$\text{SRTT} = \alpha\ \text{SRTT} + (1 - \alpha)\ \text{R}$$

onde α é um fator de nivelamento que determina a rapidez com que os valores antigos são esquecidos. Normalmente, $\alpha = 7/8$. Esse tipo de fórmula é uma média móvel ponderada exponencialmente, ou **EWMA** (**Exponentially Weighted Moving Average**), ou filtro passa-baixa, que descarta o ruído nas amostras.

Mesmo com um bom valor de *SRTT*, escolher um período de tempo de retransmissão adequado é uma questão não trivial. As implementações iniciais do TCP usavam 2*xRTT*, mas a experiência mostrou que um valor constante era muito inflexível, pois deixava de responder quando a variância subia. Em particular, modelos de enfileiramento de tráfego aleatório (ou seja, de Poisson) preveem que, quando a carga se aproxima de sua capacidade, o atraso se torna grande e altamente variável. Isso pode fazer com que o timer de retransmissão dispare e uma cópia do pacote seja retransmitida, embora o pacote original ainda esteja transitando na rede. Isso é mais provável que aconteça sob condições de alta carga, que é o pior momento para enviar pacotes adicionais para a rede.

Para resolver esse problema, Jacobson propôs tornar o valor do período de tempo sensível à variância nos tempos de ida e volta, bem como o tempo de ida e volta nivelado. Essa mudança exige registrar outra variável nivelada, *RTTVAR* (Round-Trip Time VARiation), que é atualizada usando-se a fórmula:

$$\text{RTTVAR} = \beta\ \text{RTTVAR} + (1 - \beta)\ \ |\text{SRTT} - \text{R}|$$

Esta é uma EWMA, como antes, e normalmente $\beta = 3/4$. O período de tempo de retransmissão, *RTO*, é definido como

$$\text{RTO} = \text{SRTT} + 4 \times \text{RTTVAR}$$

A escolha do fator 4 é de certa forma arbitrária, mas a multiplicação por 4 pode ser feita em um único deslocamento, e menos de 1 por cento de todos os pacotes vem atrasado com um desvio-padrão maior que quatro. Observe que a *RTTVAR* não é exatamente o mesmo que o desvio-padrão (na realidade, é o desvio da média), mas na prática é bastante próxima. O artigo de Jacobson está repleto de truques inteligentes para calcular os períodos de tempo usando apenas somas, subtrações e deslocamentos de inteiros. Essa economia agora é necessária para os hosts modernos, mas se tornou parte da cultura que permite que o TCP funcione sobre todos os tipos de dispositivos, desde supercomputadores até pequenos dispositivos. Até agora, ninguém a incluiu em um chip de RFID, mas quem sabe algum dia?

Outros detalhes de como calcular esse período de tempo, incluindo os valores iniciais das variáveis, são dados na RFC 2988. O timer de retransmissão também é mantido em um mínimo de 1 segundo, independentemente das estimativas. Esse é um valor conservador, escolhido para impedir retransmissões falsas, com base nas medições (Allman e Paxson, 1999).

Um problema que ocorre com a coleta das amostras, *R*, do tempo de ida e volta, é o que fazer quando um segmento expira seu período de tempo e é enviado novamente. Quando a confirmação chega, não fica claro se ela se refere à primeira transmissão ou a alguma outra. A decisão errada pode contaminar seriamente o período de tempo de retransmissão. Phil Karn descobriu esse problema pelo modo mais difícil. Karn é um radioamador interessado em

transmitir pacotes TCP/IP via radioamador, um meio notoriamente não confiável. Ele fez uma proposta simples: não atualizar as estimativas sobre quaisquer segmentos que tiverem sido retransmitidos. Além disso, o período de tempo é dobrado a cada retransmissão bem-sucedida, até que os segmentos passem pela primeira vez. Esse reparo é chamado **algoritmo de Karn** (Karn e Partridge, 1987). A maioria das implementações do TCP o utiliza.

O timer de retransmissão não é o único timer que o TCP utiliza. Um segundo timer é o **timer de persistência**. Ele é projetado para impedir o impasse a seguir. O receptor envia uma confirmação com um tamanho de janela 0, dizendo ao transmissor para esperar. Mais tarde, o receptor atualiza a janela, mas o pacote com a atualização se perde. Agora, o transmissor e o receptor estão esperando que o outro faça algo. Quando o timer de persistência expira, o transmissor envia uma consulta ao receptor. A resposta dessa consulta indica o tamanho da janela. Se ainda for 0, o timer de persistência é definido novamente e o ciclo se repete. Se for diferente de zero, os dados agora podem ser enviados.

O terceiro timer que algumas implementações utilizam é o **timer keepalive**. Quando uma conexão estiver ociosa por muito tempo, o timer keepalive pode expirar e fazer com que um lado verifique se o outro lado ainda está lá. Se ele não responder, a conexão é encerrada. Esse recurso é discutível, pois aumenta o overhead e pode terminar uma conexão ativa devido a uma interrupção transitória da rede.

O último timer usado em cada conexão TCP é aquele usado no estado *TIME WAIT* durante o encerramento. Ele usa o dobro do tempo de vida máximo do pacote para garantir que, quando uma conexão for fechada, todos os pacotes criados por ela terão expirado.

6.5.10 Controle de congestionamento do TCP

Deixamos uma das principais funções do TCP para o final: o controle de congestionamento. Quando a carga oferecida a qualquer rede é maior que sua capacidade, acontece um congestionamento. A Internet não é exceção a essa regra. A camada de rede detecta o congestionamento quando as filas se tornam grandes nos roteadores e tenta gerenciá-lo, mesmo que apenas descartando pacotes. Cabe à camada de transporte receber o feedback do congestionamento da camada de rede e diminuir a velocidade do tráfego que está enviando para a rede. Na Internet, o TCP desempenha o principal papel no controle do congestionamento, bem como no transporte confiável. É por isso que esse protocolo é tão especial.

Abordamos a situação geral do controle de congestionamento na Seção 6.3. Um detalhe importante foi que um protocolo de transporte usando uma lei de controle AIMD (Additive Increase Multiplicative Decrease) em resposta aos sinais de congestionamento binários da rede convergiria para uma alocação de largura de banda imparcial e eficiente. O controle de congestionamento do TCP é baseado na implementação dessa técnica usando uma janela e com perda de pacotes como sinal binário. Para fazer isso, o TCP mantém uma **janela de congestionamento** cujo tamanho é o número de bytes que o transmissor pode ter na rede a qualquer momento. A velocidade correspondente é o tamanho da janela dividido pelo tempo de ida e volta da conexão. O TCP ajusta o tamanho da janela de acordo com a regra AIMD.

Lembre-se de que a janela de congestionamento é mantida *além* da janela de controle de fluxo, que especifica o número de bytes que o receptor pode armazenar em buffer. As duas janelas são acompanhadas em paralelo, e o número de bytes que podem ser enviados é a menor das duas janelas. Assim, a janela efetiva é o menor entre o que o transmissor pensa ser o correto e o que o receptor pensa ser o correto. Se um não quer, dois não brigam. O TCP deixará de enviar dados se a janela de congestionamento ou a janela de controle de fluxo estiverem temporariamente cheias. Se o receptor disser 'envie 64 KB', mas o transmissor souber que rajadas de mais de 32 KB se acumulam na rede, ele enviará 32 KB. Por outro lado, se o receptor disser 'envie 64 KB' e o transmissor souber que rajadas de até 128 KB são enviadas sem nenhum esforço, ele enviará os 64 KB solicitados. A janela de controle de fluxo foi descrita anteriormente, e a seguir descreveremos apenas a janela de congestionamento.

O controle de congestionamento moderno foi acrescentado ao TCP em grande parte por meio dos esforços de Van Jacobson (1988). Essa é uma história fascinante. A partir de 1986, a popularidade crescente da Internet de então levou à primeira ocorrência do que passou a ser conhecido como **colapso de congestionamento**, um período prolongado durante o qual o goodput caiu bruscamente (ou seja, por um fator de mais de 100) devido ao congestionamento na rede. Jacobson e muitos outros começaram a entender o que estava acontecendo e remediaram a situação.

O reparo de alto nível que Jacobson implementou foi aproximar uma janela de congestionamento AIMD. A parte interessante, e grande parte da complexidade do controle de congestionamento do TCP, é como ele acrescentou isso a uma implementação existente sem alterar nenhum formato de mensagem, tornando-o instantaneamente implementável. Para começar, ele observou que a perda de pacote é um sinal adequado de congestionamento. Esse sinal chega um pouco tarde (quando a rede já está congestionada), mas é bastante confiável. Afinal, é difícil montar um roteador que não descarte pacotes quando está sobrecarregado. Esse fato provavelmente não mudará. Mesmo quando aparecerem memórias de terabytes para manter grandes quantidades de pacotes em buffer, provavelmente teremos redes de terabits/s para preencher essas memórias.

Porém, usar a perda de pacotes como um sinal de congestionamento depende de os erros de transmis-

são serem relativamente raros. Isso normalmente não acontece para redes sem fios, como 802.11, motivo pelo qual elas incluem seu próprio mecanismo de retransmissão na camada de enlace. Devido às retransmissões sem fios, a perda de pacotes da camada de rede normalmente é mascarada. Ela também é rara em outros enlaces, pois os fios e as fibras ópticas normalmente têm baixas taxas de erro de bit.

Todos os algoritmos TCP da Internet consideram que os pacotes perdidos são causados por congestionamento e monitoram os períodos de tempo, procurando sinais de problemas da forma como os mineiros observam seus canários. É preciso que haja um bom timer de retransmissão para detectar sinais de perda de pacotes com precisão e em tempo. Já discutimos como o timer de retransmissão do TCP inclui estimativas da média e variação nos tempos de ida e volta. Consertar esse timer, incluindo um fator de variação, foi um passo importante no trabalho de Jacobson. Dado um bom período de tempo de retransmissão, o transmissor TCP pode acompanhar o número de bytes pendentes, que estão sobrecarregando a rede. Ele simplesmente observa a diferença entre os números de sequência que são transmitidos e confirmados.

Agora, parece que nossa tarefa é fácil. Tudo o que precisamos fazer é acompanhar a janela de congestionamento, usando números de sequência e confirmação, e ajustar a janela de congestionamento usando uma regra AIMD. Como você poderia esperar, a coisa é mais complicada. Uma consideração inicial é que o modo como os pacotes são enviados para a rede, até mesmo por pequenos períodos de tempo, deve corresponder ao caminho da rede. Caso contrário, o tráfego causará congestionamento. Por exemplo, considere um host com uma janela de congestionamento de 64 KB anexada a uma Ethernet comutada de 1 Gbps. Se o host enviar a janela inteira ao mesmo tempo, essa rajada de tráfego poderá passar por uma linha ADSL lenta de 1 Mbps mais adiante no caminho. A rajada que levava meio milissegundo na linha de 1 Gbps prenderá a linha de 1 Mbps por meio segundo, tumultuando completamente protocolos como o VoIP. Esse comportamento poderia ser uma boa ideia em um protocolo projetado para causar congestionamento, mas não em um protocolo para controlá-lo.

Entretanto, acontece que podemos usar pequenas rajadas de pacotes para o nosso proveito. A Figura 6.37 mostra o que acontece quando um transmissor em uma rede rápida (o enlace de 1 Gpbs) envia uma pequena rajada de quatro pacotes para um receptor em uma rede lenta (o enlace de 1 Mbps), que é o gargalo ou a parte mais lenta do caminho. Inicialmente, os quatro pacotes atravessam o enlace o mais rapidamente quanto o transmissor puder enviá-los. No roteador, eles são enfileirados enquanto são enviados, pois leva mais tempo para enviar um pacote pelo enlace lento do que para receber o próximo pacote pelo enlace rápido. Mas a fila não é grande, pois somente um pequeno número de pacotes foi enviado ao mesmo tempo. Observe o maior tamanho dos pacotes no enlace lento. O mesmo pacote, digamos, de 1 KB, agora é maior, pois leva mais tempo para ser enviado em um enlace lento do que em um enlace rápido.

Por fim, os pacotes chegam ao receptor, onde são confirmados. Os tempos para as confirmações refletem os tempos em que os pacotes chegaram ao receptor depois de cruzar o enlace lento. Eles são espalhados em comparação com os pacotes originais no enlace rápido. À medida que essas confirmações trafegam pela rede e retornam ao transmissor, elas preservam esse tempo.

A observação principal é esta: as confirmações retornam ao transmissor aproximadamente na velocidade em que os pacotes podem ser enviados pelo enlace mais lento no caminho. Essa é exatamente a velocidade que o transmissor deseja usar. Se ele injetar novos pacotes na rede nessa velocidade, eles serão enviados na velocidade que o enlace mais lento permite, mas não ficarão enfileirados nem congestionarão nenhum roteador ao longo do caminho. Esse tempo é conhecido como **clock ACK**. Essa é uma parte essencial do TCP. Usando um clock ACK, o TCP nivela o tráfego e evita filas desnecessárias nos roteadores.

A segunda consideração é que a regra AIMD levará um tempo muito grande para alcançar um bom ponto de operação em redes rápidas se a janela de congestionamento for iniciada a partir de um tamanho pequeno. Considere uma rede modesta que pode dar suporte a 10 Mbps com um RTT de 100 ms. A janela de congestionamento apropriada é o produto largura de banda-atraso, que é 1 Mbit ou 100 pacotes de 1.250 bytes cada um. Se a janela de congestionamento começar em 1 pacote e aumentar em 1 pacote a

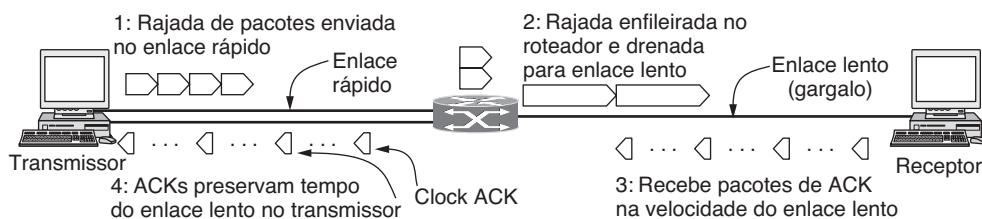



Figura 6.37 | Uma rajada de pacotes de um transmissor e o clock ACK retornando.

cada RTT, ela será de 100 RTTs ou 10 segundos antes que a conexão esteja rodando na velocidade correta. Esse é um longo tempo a esperar só para chegar à velocidade certa para uma transferência. Poderíamos reduzir esse tempo inicial começando com uma janela inicial maior, digamos, de 50 pacotes. Mas essa janela seria muito grande para enlaces lentos ou curtos. Isso causaria congestionamento se fosse usado tudo ao mesmo tempo, conforme acabamos de descrever.

Em vez disso, a solução que Jacobson achou para lidar com essas duas considerações é uma mistura de aumento linear e multiplicativo. Quando a conexão é estabelecida, o transmissor inicia a janela de congestionamento em um valor inicial pequeno de, no máximo, quatro segmentos; os detalhes são descritos na RFC 3390, e o uso de quatro segmentos é um aumento de um segmento a partir do valor inicial, com base na experiência. O transmissor então envia a janela inicial. Os pacotes levarão um tempo de ida e volta para ser confirmados. Para cada segmento que é confirmado antes que o timer de retransmissão expire, o transmissor soma os bytes correspondentes a um segmento na janela de congestionamento. Além disso, quando esse segmento tiver sido confirmado, existe um a menos na rede. O resultado é que cada segmento confirmado permite que mais dois sejam enviados. A janela de congestionamento está dobrando a cada tempo de ida e volta.

Esse algoritmo é chamado **partida lenta**, mas não é nada lento — ele tem crescimento exponencial —, exceto em comparação com o algoritmo anterior, que permitia que uma janela de controle de fluxo inteira fosse enviada ao mesmo tempo. A partida lenta pode ser vista na Figura 6.38. No primeiro tempo de ida e volta, o transmissor injeta um pacote na rede (e o receptor recebe um pacote). Dois pacotes são enviados no próximo tempo de ida e volta, depois quatro pacotes no terceiro tempo de ida e volta.

A partida lenta funciona bem para uma faixa de velocidades de enlace e tempos de ida e volta e utiliza um clock ACK para combinar a velocidade do transmissor com o caminho na rede. Dê uma olhada no modo como as confirmações retornam do transmissor ao receptor na Figura 6.38. Quando o transmissor recebe uma confirmação, ele aumenta a janela de congestionamento em um e imediatamente envia dois pacotes para a rede. (Um pacote é o aumento de um; o outro pacote é um substituto para o pacote que foi confirmado e saiu da rede. Em todo o tempo, o número de pacotes confirmados é dado pela janela de congestionamento.) Porém, esses dois pacotes não necessariamente chegarão ao receptor tão próximos um do outro quanto foram enviados. Por exemplo, suponha que o transmissor esteja em uma Ethernet de 100 Mbps. Cada pacote de 1.250 bytes leva 100 μs para ser enviado. Assim, o atraso entre os pacotes pode ser tão pequeno quanto 100 μs. A situação muda se esses pacotes atravessarem um enlace ADSL de 1 Mbps durante o caminho. Agora, são gastos 10 ms para enviar o mesmo pacote. Isso significa que o espaçamento mínimo entre os dois pacotes aumentou por um fator de 100. A menos que os pacotes tenham que esperar juntos em uma fila de um enlace mais adiante, o espaçamento continuará sendo grande.

Na Figura 6.38, esse esforço é mostrado impondo-se um espaçamento mínimo entre os pacotes de dados que chegam ao receptor. O mesmo espaçamento é mantido quando o receptor envia confirmações e, consequentemente, quando o transmissor recebe as confirmações. Se o caminho da rede for lento, as confirmações chegarão lentamente (após um atraso de um RTT). Se o caminho na rede for rápido, as confirmações chegarão rapidamente (novamente, após o RTT). Tudo o que o transmissor precisa fazer é seguir o tempo do clock ACK enquanto injeta novos pacotes, que é o que a partida lenta faz.

Como a partida lenta causa crescimento exponencial, em algum momento (mas não muito depois) ele enviará muitos pacotes para a rede muito rapidamente. Quando isso acontece, as filas se acumulam na rede. Quando as filas estiverem cheias, um ou mais pacotes serão perdidos. Depois que isso acontecer, o transmissor TCP expirará seu período de tempo quando uma confirmação não chegar em tempo. Na Figura 6.38, há evidência de um crescimento muito rápido da partida lenta. Depois de três RTTs, quatro pacotes estão na rede. Eles usam um RTT inteiro para chegar ao receptor. Ou seja, uma janela de congestionamento de quatro pacotes tem o tamanho certo para essa conexão. Porém, à medida que esses pacotes são confirmados, a partida lenta continua a aumentar a janela de congestionamento, alcançando oito pacotes em outro RTT. Apenas quatro deles podem alcançar o receptor em um RTT, não importa quantos sejam enviados; ou seja, o canal da rede está cheio. Pacotes adicionais colocados na rede pelo transmissor se acumularão nas filas do roteador, pois não podem ser entregues ao receptor com rapidez suficiente. Logo, haverá congestionamento e perda de pacotes.

Para conservar a partida lenta sob controle, o transmissor mantém um limite para a conexão chamado **limite**

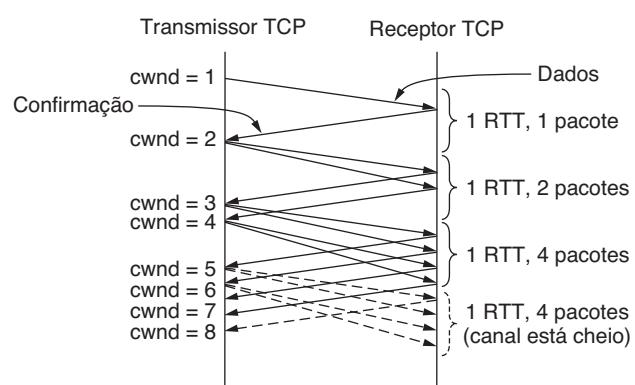



Figura 6.38 | Partida lenta de uma janela de congestionamento inicial de um segmento.

de partida lenta. Inicialmente, isso é definido com um valor arbitrariamente alto, com o tamanho da janela de controle de fluxo, de modo que não limitará a conexão. O TCP continua aumentando a janela de congestionamento na partida lenta até que um período de tempo expire ou a janela de congestionamento ultrapasse o limite (ou a janela do receptor esteja cheia).

Sempre que uma perda de pacote é detectada, por exemplo, por um período de tempo expirado, o limite de partida lenta é definido para a metade da janela de congestionamento e o processo inteiro é reiniciado. A ideia é que a janela atual é muito grande, pois causou um congestionamento anteriormente que só agora foi detectado por um período de tempo expirado. Metade da janela, que foi usada com sucesso em um momento anterior, provavelmente é uma melhor estimativa para uma janela de congestionamento que está próxima da capacidade do caminho, mas não causará perda. Em nosso exemplo da Figura 6.38, aumentar a janela de congestionamento para oito pacotes pode causar perda, enquanto a janela de congestionamento de quatro pacotes no RTT anterior foi o valor correto. A janela de congestionamento, então, retorna ao seu valor mínimo inicial e a partida lenta é retomada.

Sempre que o limite de partida lenta é ultrapassado, o TCP passa de partida lenta para aumento aditivo. Nesse modo, a janela de congestionamento é aumentada em um segmento a cada tempo de ida e volta. Assim como a partida lenta, isso normalmente é implementado com um aumento para cada segmento que é confirmado, ao invés de um aumento único por RTT. Sejam a janela de congestionamento *cwnd* e o tamanho máximo de segmento *MSS*. Uma aproximação comum é aumentar *cwnd* em (*MSS* × *MSS*)/*cwnd* para cada um dos *cwnd*/*MSS* pacotes que podem ser confirmados. Esse aumento não precisa ser rápido. A ideia completa é que uma conexão TCP gasta muito tempo com sua janela de congestionamento próxima do valor ideal — não tão pequeno para que a vazão seja baixa, nem tão grande para que ocorra congestionamento.

O aumento aditivo aparece na Figura 6.39 para a mesma situação da partida lenta. Ao final de cada RTT, a janela de congestionamento do transmissor terá crescido o suficiente para que possa injetar um pacote adicional na rede. Em comparação com a partida lenta, a taxa de crescimento linear é muito mais lenta. Ela faz pouca diferença para janelas de congestionamento pequenas, como acontece aqui, mas uma diferença grande no tempo gasto para aumentar a janela de congestionamento para 100 segmentos, por exemplo.

Há mais uma coisa que podemos fazer para melhorar o desempenho. O defeito no esquema até aqui é esperar por um período de tempo. Os períodos de tempo são relativamente longos, pois precisam ser conservadores. Após a perda de um pacote, o receptor não pode confirmar além dele, de modo que o número de confirmação permanecerá fixo, e o transmissor não poderá enviar novos pacotes para a rede, pois sua janela de congestionamento permanece cheia. Essa condição pode continuar por um período relativamente longo, até que o timer seja disparado e o pacote perdido seja retransmitido. Nesse estágio, o TCP inicia novamente a partida lenta.

Há um modo rápido para o transmissor reconhecer que um de seus pacotes se perdeu. Quando os pacotes além do pacote perdido chegam ao receptor, eles disparam confirmações que retornam ao transmissor. Essas confirmações contêm o mesmo número de confirmação. Elas são chamadas **confirmações duplicadas**. Toda vez que o transmissor recebe uma confirmação duplicada, é provável que outro pacote tenha chegado ao receptor e o pacote perdido ainda não tenha aparecido.

Como os pacotes podem tomar caminhos diferentes pela rede, eles podem chegar fora de ordem. Isso disparará confirmações duplicadas, embora nenhum pacote tenha sido perdido. Contudo, isso é raro na Internet na maior parte do tempo. Quando existe reordenação por vários caminhos, os pacotes recebidos normalmente não são reordenados em demasia. Assim, o TCP, de uma forma um tanto arbitrária, considera que três confirmações duplicadas significam que um pacote foi perdido. A identidade do pacote perdido também pode ser deduzida pelo número de confirmação. Ele é o próximo na sequência. Esse pacote pode então ser retransmitido imediatamente, antes que o período de tempo para retransmissão expire.

Essa heurística é chamada **retransmissão rápida**. Depois que ela é disparada, o limite de partida lenta ainda está definido como metade da janela de congestionamento atual, assim como em um período de tempo. A partida lenta pode ser reiniciada definindo-se a janela de congestionamento para um pacote. Com essa janela de congestionamento, um novo pacote será enviado após um tempo de ida e volta que é gasto para confirmar o pacote retransmitido junto

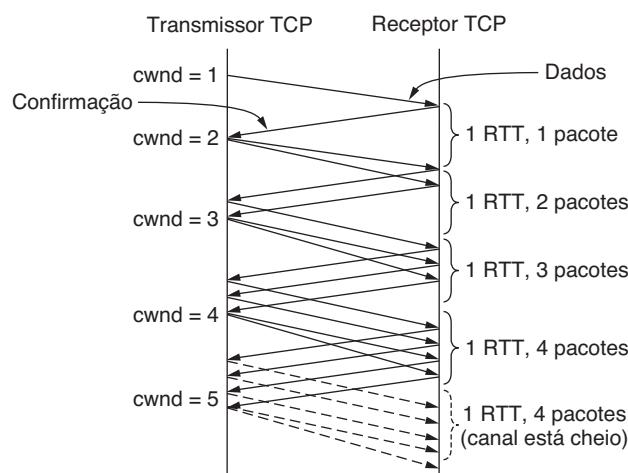



Figura 6.39 | Aumento aditivo a partir de uma janela de congestionamento inicial de um segmento.

com todos os dados que foram enviados antes que a perda fosse detectada.

Uma ilustração do algoritmo de congestionamento que criamos até aqui aparece na Figura 6.40. Essa versão do TCP é chamada TCP Tahoe, devido à versão 4.2BSD Tahoe em que ela foi incluída, em 1988. O tamanho máximo do segmento aqui é 1 KB. Inicialmente, a janela de congestionamento era de 64 KB, mas houve um período de tempo expirado, de modo que o limite é definido como 32 KB e a janela de congestionamento para 1 KB para a **transmissão 0**. A janela de congestionamento cresce exponencialmente até atingir o limite (32 KB). A janela aumenta toda vez que uma nova confirmação chega, e não de forma contínua, o que leva ao padrão descontínuo em escada. Após o limite ser ultrapassado, a janela cresce linearmente. Ela aumenta em um segmento a cada RTT.

As transmissões na rodada 13 não têm sorte (elas deveriam saber disso), e uma delas se perde na rede. Isso é detectado quando chegam três confirmações duplicadas. Nesse momento, o pacote perdido é retransmitido, o limite é definido para metade da janela atual (no momento, 40 KB, de modo que a metade é 20 KB) e a partida lenta é iniciada novamente. Reiniciar com uma janela de congestionamento de um pacote exige um tempo de ida e volta para todos os dados previamente transmitidos saírem da rede e ser confirmados, incluindo o pacote retransmitido. A janela de congestionamento cresce com a partida lenta, como fazia anteriormente, até alcançar o novo limite de 20 KB. Nesse momento, o crescimento se torna linear novamente. Ele continuará dessa forma até que outra perda de pacote seja detectada por confirmações duplicadas ou até que um período de tempo seja expirado (ou a janela do receptor chegue ao extremo).

O TCP Tahoe (que incluía bons timers de retransmissão) ofereceu um algoritmo de controle de congestionamento funcional, que resolveu o problema do colapso do congestionamento. Jacobson observou que é possível fazer ainda melhor. No momento da retransmissão rápida, a conexão está trabalhando com uma janela de congestionamento muito grande, mas ainda está trabalhando com um clock ACK funcional. Toda vez que outra confirmação duplicada chega, é provável que outro pacote tenha saído da rede. Usando confirmações duplicadas para contar os pacotes na rede, é possível permitir que alguns pacotes saiam da rede e continuem a enviar um novo pacote para cada confirmação duplicada adicional.

Recuperação rápida é a heurística que implementa esse comportamento. Esse é um modo temporário que visa a manter o clock ACK funcionando com uma janela de congestionamento que é o novo limite, ou metade do valor da janela de congestionamento no momento da retransmissão rápida. Para fazer isso, confirmações duplicadas são contadas (incluindo as três que dispararam a retransmissão rápida) até que o número de pacotes na rede tenha caído para o novo limite. Isso leva cerca de metade de um tempo de ida e volta. Daí em diante, um novo pacote pode ser enviado para cada confirmação duplicada recebida. Um tempo de ida e volta após a retransmissão rápida e o pacote perdido terá sido confirmado. Nesse momento, o fluxo de confirmações duplicadas terminará e o modo de recuperação rápida será encerrado. A janela de congestionamento será definida para o novo limite de partida lenta e crescerá pelo aumento linear.

O resultado dessa heurística é que o TCP evita a partida lenta, exceto quando a conexão é iniciada e quando um período de tempo expira. O último também pode acontecer quando mais de um pacote é perdido e a retransmissão rápida não se recupera adequadamente. Em vez de partidas lentas repetidas, a janela de congestionamento de uma conexão em execução segue um padrão **dente de serra** de aumento aditivo (por um segmento a cada RTT) e diminuição multiplicativa (pela metade em

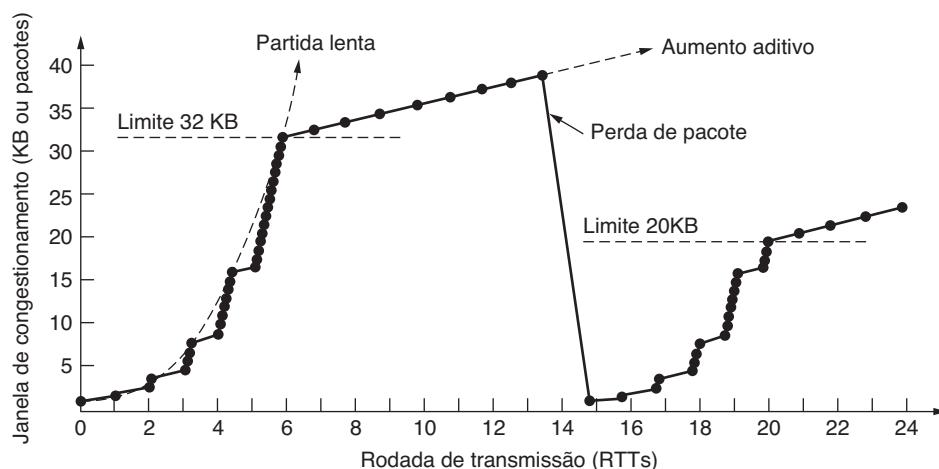



Figura 6.40 | Partida lenta seguida pelo aumento aditivo no TCP Tahoe.

um RTT). Essa é exatamente a regra AIMD que procuramos implementar.

Esse comportamento dente de serra aparece na Figura 6.41. Ele é produzido pelo TCP Reno, que tem o nome da versão 4.3BSD Reno de 1990, na qual foi incluído. O TCP Reno é basicamente o TCP Tahoe com recuperação rápida. Após uma partida lenta inicial, a janela de congestionamento sobe linearmente até que uma perda de pacote seja detectada pelas confirmações duplicadas. O pacote perdido é retransmitido e a recuperação rápida é usada para manter o clock ACK funcionando até que a retransmissão seja confirmada. Nesse momento, a janela de congestionamento é retomada a partir do novo limite de partida lenta, ao invés de 1. Esse comportamento continua indefinidamente, e a conexão gasta a maior parte do tempo com sua janela de congestionamento próxima do valor ideal do produto largura de banda-atraso.

O TCP Reno, com seus mecanismos para ajustar a janela de congestionamento, formou a base para o controle de congestionamento TCP por mais de duas décadas. A maior parte das mudanças nos anos intervenientes ajustou esses mecanismos de algumas formas, por exemplo, alterando as escolhas da janela inicial e removendo diversas ambiguidades. Algumas melhorias foram feitas para a recuperação de duas ou mais perdas em uma janela de pacotes. Por exemplo, a versão TCP NewReno usa um avanço parcial do número de confirmação após uma transmissão encontrar e reparar outra perda (Hoe, 1996), conforme descrito na RFC 3782. Desde meados da década de 90, surgiram diversas variações que seguem os princípios que descrevemos, mas que usam leis de controle ligeiramente diferentes. Por exemplo, o Linux usa uma variante chamada CUBIC TCP (He et al., 2008) e o Windows inclui uma variante chamada TCP composto (Tan et al., 2006).

Duas mudanças maiores também afetaram as implementações do TCP. Primeiro, grande parte da complexidade do TCP vem da dedução, por um fluxo de confirmações duplicadas, de quais pacotes chegaram e quais pacotes se perderam. O número de confirmação acumulativo não oferece essa informação. Um reparo simples é o uso de **SACK** (**Selective ACKnowledgements**), que lista até três intervalos de bytes que foram recebidos. Com essa informação, o transmissor pode decidir mais diretamente quais pacotes retransmitir e acompanhar os pacotes a caminho para implementar a janela de congestionamento.

Quando transmissor e receptor estabelecem uma conexão, cada um deles envia a opção do TCP *SACK permitido* para sinalizar que eles entendem as confirmações seletivas. Quando o SACK é ativado para uma conexão, ela funciona como mostra a Figura 6.42. Um receptor usa o campo de *Número de confirmação* do TCP normalmente, como uma confirmação acumulativa do byte mais alto na ordem que foi recebido. Ao receber o pacote 3 fora de ordem (porque o pacote 2 se perdeu), ele envia uma *opção SACK* para os dados recebidos junto com a confirmação acumulativa (duplicada) para o pacote 1. A *opção SACK* oferece os intervalos de bytes que foram recebidos acima do número dado pela confirmação acumulativa. O primeiro intervalo é o pacote que disparou a confirmação duplicada. Os próximos intervalos, se houver, são blocos mais antigos. Até três intervalos normalmente são usados. Quando o pacote 6 é recebido, dois intervalos de bytes SACK são usados para indicar que o pacote 6 e os pacotes 3 e 4 foram recebidos, além de todos os pacotes até o pacote 1. Pela informação em cada *opção SACK* que recebe, o transmissor pode decidir quais pacotes retransmitir. Nesse caso, a retransmissão dos pacotes de 2 a 5 seria uma boa ideia.

O SACK é informação estritamente consultiva. A detecção real de perda usando confirmações duplicadas e

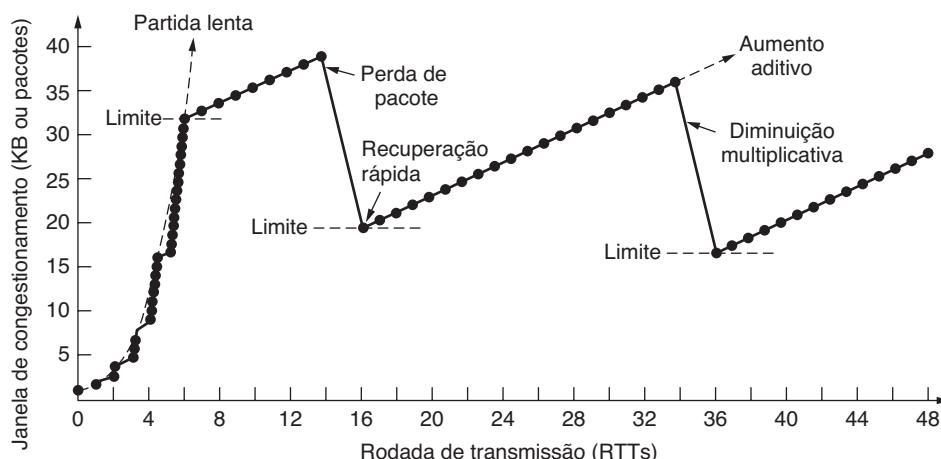



Figura 6.41 | Recuperação rápida e padrão dente de serra do TCP Reno.

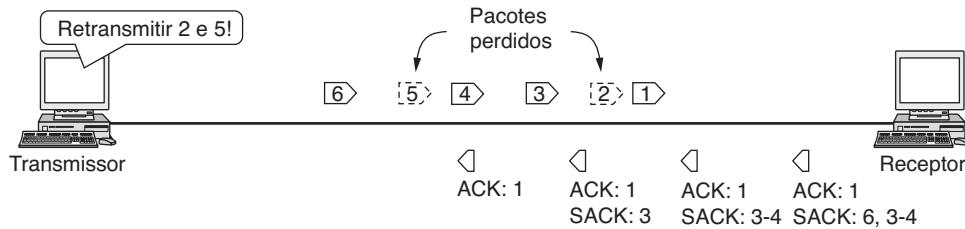



Figura 6.42 | Confirmações seletivas.

ajustes na janela de congestionamento prossegue exatamente como antes. Porém, com SACK, o TCP pode se recuperar mais facilmente de situações em que vários pacotes se perdem aproximadamente ao mesmo tempo, pois o transmissor TCP sabe quais pacotes não foram recebidos. O SACK agora é muito utilizado. Ele é descrito na RFC 2883, e o controle de congestionamento TCP usando SACK é descrito na RFC 3517.

A segunda mudança é o uso de ECN (Explicit Congestion Notification), além da perda de dados como um sinal de congestionamento. ECN é um mecanismo da camada IP para notificar os hosts quanto ao congestionamento que descrevemos na Seção 5.3.4. Com ele, o receptor TCP pode receber sinais de congestionamento do IP.

O uso de ECN é ativado para uma conexão TCP quando transmissor e receptor indicam que são capazes de usar ECN definindo os bits *ECE* e *CWR* durante o estabelecimento da conexão. Se o bit ECN for usado, cada pacote que transporta um segmento TCP é marcado no cabeçalho IP para mostrar que pode transportar um sinal ECN. Os roteadores que dão suporte ao mecanismo ECN definirão um sinal de congestionamento nos pacotes que podem transportar flags ECN quando o congestionamento estiver se aproximando, em vez de descartar esses pacotes após a ocorrência do congestionamento.

O receptor TCP é informado se qualquer pacote que chega transportar um sinal de congestionamento ECN. O receptor, então, usa a flag *ECE* (ECN-Echo) para sinalizar ao transmissor TCP que seus pacotes sofreram congestionamento. O transmissor diz ao receptor que ele ouviu o sinal por meio da flag *CWR* (Congestion Window Reduced).

O transmissor TCP reage a essas notificações de congestionamento exatamente da mesma maneira como faz com a perda de pacotes que é detectada por confirmações duplicadas. Porém, a situação é estritamente melhor. O congestionamento foi detectado e nenhum pacote foi prejudicado de forma alguma. O mecanismo ECN é descrito na RFC 3168. Ele exige suporte do host e do roteador, e ainda não é muito usado na Internet.

Para obter mais informações sobre o conjunto completo de comportamentos de controle de congestionamento implementados no TCO, consulte a RFC 5681.

6.5.11 O futuro do TCP

Como força motriz da Internet, o TCP tem sido usado em muitas aplicações e estendido com o passar do tempo para oferecer bom desempenho em topologias de redes cada vez maiores. Muitas versões são implantadas com implementações ligeiramente diferentes dos algoritmos clássicos que descrevemos, especialmente para controle de congestionamento e robustez contra ataques. É provável que o TCP continue a evoluir com a Internet. Nesta seção, vamos mencionar dois aspectos em particular.

O primeiro é que o TCP não oferece a semântica de transporte que todas as aplicações desejam. Por exemplo, algumas aplicações desejam enviar mensagens ou registros cujos limites precisam ser preservados. Outras aplicações trabalham com um grupo de conversações relacionadas, como um navegador Web que transfere vários objetos a partir do mesmo servidor. Ainda outras aplicações desejam melhorar o controle sobre os caminhos na rede que elas utilizam. O TCP com suas interfaces de soquetes-padrão não atende muito bem a essas necessidades. Basicamente, a aplicação tem o peso de lidar com qualquer problema não resolvido pelo TCP. Isso gerou propostas para novos protocolos que oferecessem uma interface ligeiramente diferente. Dois exemplos são SCTP (Stream Control Transmission Protocol), definido na RFC 4960, e SST (Structured Stream Transport) (Ford, 2007). Porém, toda vez que alguém propõe mudar algo que tenha funcionado tão bem por tanto tempo, há sempre uma imensa batalha entre estes dois lados: 'os usuários estão exigindo mais recursos' e 'se não quebrou, não conserte'.

O segundo é o controle de congestionamento. Você poderia esperar que esse fosse um problema resolvido após nossas deliberações e os mecanismos que foram desenvolvidos com o tempo. Não é bem assim. A forma de controle de congestionamento TCP que descrevemos, que é mais usada, é baseada em perdas de pacote como um sinal de congestionamento. Quando Padhye et al. (1998) modelaram a vazão do TCP com base no padrão dente de serra, eles descobriram que a taxa de perda de pacotes precisa cair rapidamente com o aumento de velocidade. Para alcançar uma vazão de 1 Gbps com um tempo de ida e volta de 100 ms e pacotes de 1.500 bytes,

um pacote pode ser perdido aproximadamente a cada 10 minutos. Essa é uma taxa de perda de pacote de 2×10^{-8}, que é incrivelmente pequena. Isso é muito pouco frequente para servir como um bom sinal de congestionamento, e qualquer outra fonte de perda (por exemplo, taxas de erro de transmissão de pacote de 10^{-7}) pode facilmente dominá-la, limitando a vazão.

Esse relacionamento não foi problema no passado, mas as redes estão se tornando cada vez mais rápidas, levando muitas pessoas a revisar o controle de congestionamento. Uma possibilidade é usar um controle de congestionamento alternativo, em que o sinal não é nenhuma perda de pacote. Mostramos vários exemplos na Seção 6.2. O sinal poderia ser o tempo de ida e volta, que cresce quando a rede se torna congestionada, conforme usado pelo FAST TCP (Wei et al., 2006). Outras técnicas também são possíveis, e o tempo mostrará qual é a melhor.

6.6 Questões de desempenho

As questões referentes ao desempenho são muito importantes nas redes de computadores. Quando centenas de milhares de computadores estão interconectados, são comuns interações complexas que trazem consequências imprevistas. Com frequência, essa complexidade resulta em um fraco desempenho, cujas razões todos desconhecem. Nas próximas seções, examinaremos várias questões relacionadas ao desempenho das redes, a fim de constatar que tipos de problemas existem e o que pode ser feito para solucioná-los.

Infelizmente, compreender o desempenho de uma rede é mais uma arte que uma ciência. Pouco do que existe em termos de teoria é realmente útil na prática. O que podemos fazer e oferecer são regras práticas que aprendemos com a experiência e apresentar exemplos reais. Deixamos essa discussão para depois do estudo da camada de transporte nas redes TCP porque o desempenho que as aplicações recebem depende do desempenho combinado das camadas de transporte, rede e enlace, e para podermos usar o TCP como exemplo em vários lugares.

Nas próximas seções, estudaremos seis aspectos do desempenho das redes:

1. Problemas de desempenho.
2. Medição do desempenho da rede.
3. Projeto de host para redes rápidas.
4. Processamento rápido de segmentos.
5. Compactação de cabeçalho.
6. Protocolos para redes longas e saturadas.

Esses aspectos consideram o desempenho da rede tanto no host quanto pela rede, e à medida que as redes aumentam em velocidade e tamanho.

6.6.1 Problemas de desempenho em redes de computadores

Alguns problemas de desempenho, como o congestionamento, são causados pela sobrecarga temporária de recursos. Se, de repente, um roteador receber um tráfego maior do que é capaz de manipular, haverá um congestionamento e uma queda de desempenho. Estudamos o congestionamento em detalhes neste capítulo e no anterior.

O desempenho também é prejudicado quando há um desequilíbrio nos recursos estruturais. Por exemplo, se uma linha de comunicação de gigabits estiver associada a um PC com poucos recursos, o fraco host não será capaz de processar os pacotes recebidos com a rapidez necessária, e alguns deles serão perdidos. Esses pacotes serão retransmitidos, aumentando o atraso, desperdiçando largura de banda e geralmente reduzindo o desempenho.

As sobrecargas também podem ser disparadas de forma sincronizada. Por exemplo, se um segmento contiver um parâmetro inadequado (como a porta a que ele se destina), em muitos casos o receptor, muito solícito, retornará uma notificação de erro. Agora, imagine o que poderia acontecer se um segmento inadequado fosse transmitido por broadcast para 10 mil máquinas; cada uma delas poderia enviar de volta uma mensagem de erro. Isso causaria um **congestionamento de broadcasts** que poderia paralisar a rede. O UDP sofreu com esse problema até que o protocolo ICMP fosse alterado para impedir que os hosts respondessem a erros nos segmentos UDP enviados a endereços de broadcast. As redes sem fios, particularmente, precisam ter cuidado para evitar respostas de broadcast não verificadas, pois o broadcast ocorre naturalmente e a largura de banda sem fios é limitada.

Um segundo exemplo de sobrecarga sincronizada é o que acontece depois de uma falha no fornecimento de energia elétrica. Quando a energia retorna, todas as máquinas reiniciam ao mesmo tempo. Uma sequência de reinício típica poderia exigir o acesso a um servidor qualquer (DHCP), para confirmar a identidade do usuário, e depois a algum servidor de arquivos para obter uma cópia do sistema operacional. Se centenas de máquinas em um centro de dados fizerem isso ao mesmo tempo, o servidor provavelmente interromperá seu funcionamento devido à sobrecarga.

Mesmo na ausência de sobrecargas sincronizadas e quando há recursos suficientes disponíveis, pode ocorrer um desempenho fraco devido à falta de ajuste do sistema. Por exemplo, em uma máquina com uma CPU potente e muita memória, mas com pouca memória alocada em buffers, o controle de fluxo atrasará o recebimento do segmento e limitará o desempenho. Esse foi um problema para muitas conexões TCP quando a Internet se tornou mais rápida, mas o tamanho-padrão da janela de controle de fluxo permaneceu fixo em 64 KB.

Outra questão relacionada ao ajuste é definir os períodos de tempo de forma correta. Quando um segmento é

enviado, em geral um timer é ativado para evitar sua perda. Se o período de tempo for muito curto, haverá retransmissões desnecessárias, aumentando o volume do tráfego. Se, ao contrário, o intervalo for muito longo, ocorrerão atrasos desnecessários após a perda de um segmento. Outros parâmetros ajustáveis incluem o tempo de espera por dados para enviar uma confirmação por piggyback antes de decidir enviar uma confirmação separada, e ainda o número de retransmissões antes de uma desistência.

Outro problema de desempenho que ocorre com as aplicações em que o tempo de transmissão tem importância fundamental, como sinais de áudio e vídeo, é o jitter. Atrasos de transmissão curtos também são necessários. Conseguir consistentemente atrasos curtos exige uma engenharia cuidadosa da carga na rede, suporte para qualidade de serviço no enlace e nas camadas de rede, ou ambos.

6.6.2 Medição do desempenho da rede

Quando uma rede tem baixo desempenho, em geral os usuários reclamam com seus administradores, exigindo melhorias. Para melhorar o desempenho, os operadores devem primeiro descobrir exatamente o que está acontecendo. Para isso, os operadores precisarão fazer medições. Nesta seção, veremos as medições do desempenho da rede. O estudo a seguir se baseia no trabalho de Mogul (1993).

As medições podem ser feitas de várias maneiras e em diferentes pontos da rede (tanto nos elementos físicos quanto na pilha de protocolos). O tipo mais elementar de medição consiste em ativar um timer ao iniciar um procedimento e usá-lo com a finalidade de verificar o tempo necessário para concluir essa atividade. Por exemplo, é fundamental saber quanto tempo é necessário para um segmento ser confirmado. Outras medições são realizadas com contadores que registram a frequência com que algum evento aconteceu (por exemplo, o número de segmentos perdidos). Por último, sempre há um interesse em saber a quantidade de algo, como o número de bytes processados em determinado intervalo de tempo.

A medição dos parâmetros e do desempenho das redes tem muitas armadilhas potenciais. Apresentaremos a seguir uma lista com algumas delas. Qualquer tentativa sistemática de medir o desempenho das redes deve ter o cuidado de evitar essas armadilhas.

Certifique-se de que o tamanho da amostra é grande o bastante

Não meça o tempo necessário para enviar um segmento, mas repita a medição, digamos, um milhão de vezes e tire a média. Efeitos de partida, como a NIC 802.16 ou modem a cabo obtendo uma reserva de largura de banda após um período ocioso, podem atrasar o primeiro segmento, e o enfileiramento introduz a variabilidade. Com uma grande amostra, será possível reduzir a incerteza (a variação) na medição da média e do desvio-padrão. Essa incerteza pode ser calculada com o emprego de fórmulas estatísticas-padrão.

Certifique-se de que as amostras são representativas

O ideal é que toda a sequência de um milhão de medições seja repetida em dias e horários distintos para que o efeito de diferentes cargas do sistema sobre a quantidade medida possa ser verificado. Por exemplo, medições de congestionamento terão pouca utilidade se forem feitas nos momentos em que não há congestionamento. Algumas vezes, talvez os resultados não pareçam intuitivos a princípio, como no caso de um grande congestionamento que acontece às 11h e às 13h, mas não ao meio-dia (quando todos os usuários saem para almoçar).

Com as redes sem fios, o local é uma variável importante, devido à propagação do sinal. Até mesmo um nó de medição colocado próximo de um cliente sem fios pode não observar os mesmos pacotes que o cliente, devido a diferenças nas antenas. É melhor fazer medições do cliente sem fios em estudo para ver o que ele está vendo. Se isso não funcionar, é possível usar técnicas para combinar as medições sem fios tomadas em diferentes posições de superioridade para obter uma imagem completa do que está acontecendo (Mahajan et al., 2006).

O caching pode ser destruído com as medições

Repetir uma medição muitas vezes retornará uma resposta inesperadamente rápida se os protocolos usarem mecanismos de caching. Por exemplo, a busca de uma página Web ou a pesquisa de um nome de DNS (para encontrar o endereço IP) podem envolver uma troca de rede pela primeira vez, e depois retornar a resposta de um cache local sem enviar nenhum pacote pela rede. Os resultados dessa medição são praticamente inúteis (a menos que você queira medir o desempenho do cache).

O buffering pode ter um efeito semelhante. Os testes de desempenho do TCP/IP têm sido conhecidos por relatar que o UDP pode alcançar um desempenho substancialmente mais alto do que a rede permite. Como isso ocorre? Uma chamada UDP normalmente retorna o controle assim que a mensagem tiver sido aceita pelo núcleo e acrescentada à fila de transmissão. Se houver espaço suficiente em buffer, a temporização de mil chamadas UDP não significa que todos os dados foram enviados. A maioria deles ainda pode estar no núcleo, mas o programa de teste de desempenho acha que eles foram transmitidos.

É aconselhável ter cuidado absoluto para que você entenda como os dados podem ser mantidos em cache e armazenados em buffer como parte de uma operação da rede.

Certifique-se de que nenhum evento inesperado está ocorrendo durante os testes

Fazer medições ao mesmo tempo em que algum usuário decidiu realizar uma conferência de vídeo por sua rede normalmente dará resultados diferentes do que se não houvesse conferência de vídeo. É melhor executar testes em uma rede ociosa e criar você mesmo a carga de trabalho

inteira. Entretanto, até mesmo essa técnica apresenta armadilhas. Embora você possa imaginar que ninguém estará usando a rede às 3 horas da manhã, esse pode ser exatamente o momento em que o programa de backup automático começa a copiar o conteúdo de todos os discos para uma fita. Além disso, talvez haja um tráfego intenso para as páginas da World Wide Web de que você mais gosta, devido às diferenças de fuso horário.

As redes sem fios são desafiadoras em relação a isso, pois normalmente não é possível separá-las de todas as fontes de interferência. Mesmo que não haja outras redes sem fios enviando tráfego nas proximidades, alguém pode fazer pipoca de microondas e inadvertidamente causar interferência que degrada o desempenho da rede 802.11. Por esses motivos, é uma boa prática monitorar a atividade geral da rede de modo que você possa pelo menos observar quando algo inesperado acontecer.

Tenha cuidado ao usar o clock do computador

Os clocks dos computadores funcionam incrementando algum contador a intervalos regulares. Por exemplo, um timer de milissegundos acrescenta uma unidade a um contador a cada milissegundo. A utilização de um timer para medir um evento que dura menos de 1 milissegundo não é impossível, mas requer algum cuidado. É claro que alguns computadores têm clocks mais precisos, mas sempre existem também eventos mais curtos para medir. Observe que os clocks nem sempre são tão acurados quanto à precisão com que o tempo é retornado quando eles são lidos.

Por exemplo, quando se mede o tempo necessário para fazer uma conexão TCP, o clock do sistema (digamos, em milissegundos) deve ser lido na entrada do código da camada de transporte e na saída desse código. Se o tempo de conexão real for de 300 μs, o valor da diferença entre as duas leituras será 0 ou 1, ambos errados. Entretanto, se a medição for repetida um milhão de vezes e o total das medições for somado e dividido por um milhão, o tempo médio terá uma precisão melhor que 1 μs.

Tenha cuidado para não extrapolar os resultados

Suponha que você tenha realizado medições utilizando cargas de rede simuladas que variam de 0 (inativa) a 0,4 (40 por cento de sua capacidade). Por exemplo, o tempo de resposta para enviar um pacote VoIP por uma rede 802.11 poderia ser como mostram os pontos de dados e a linha contínua que passa por eles na Figura 6.43. Talvez seja interessante extrapolar os resultados linearmente, como mostra a linha pontilhada. Entretanto, muitos resultados de enfileiramento envolvem um fator $1/(1-\rho)$, onde ρ é a carga; portanto, os valores verdadeiros devem ser mais parecidos com os valores da linha tracejada, que aumentam com rapidez muito maior que a linear quando a carga se torna alta. Ou seja, cuidado com os efeitos de disputa que se tornam muito mais pronunciados em carga alta.

6.6.3 Projeto de host para redes rápidas

Medições e ajustes podem melhorar o desempenho consideravelmente, mas não podem substituir um bom projeto. Uma rede mal projetada pode ser melhorada, mas só até certo ponto. Daí em diante, ela precisa ser totalmente refeita.

Nesta seção, apresentaremos algumas regras práticas para implementação de software de protocolos de rede em hosts. Surpreendentemente, a experiência mostra que isso normalmente é um gargalo de desempenho em redes

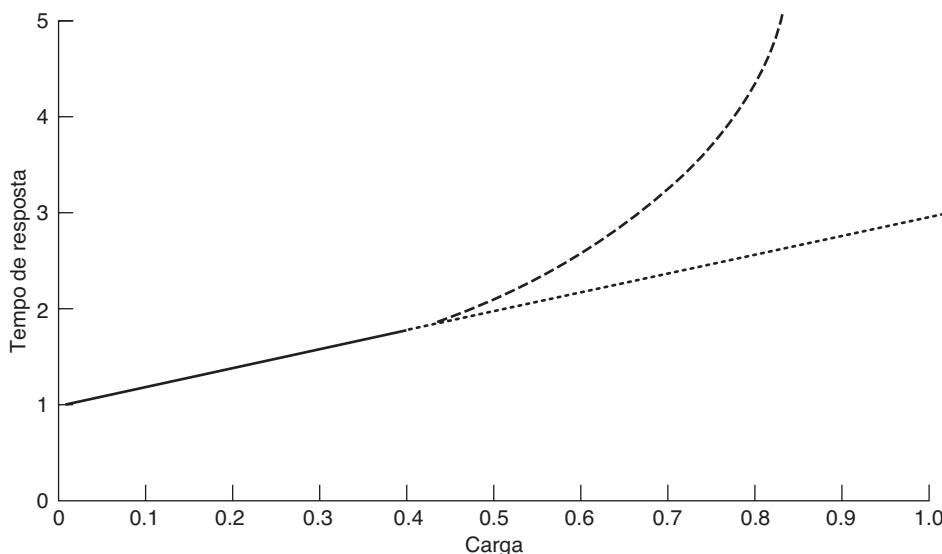



Figura 6.43 | Resposta em função da carga.

que poderiam ser rápidas, por dois motivos. Primeiro, as NICs (placas de interface de rede) e os roteadores já foram projetados (com suporte de hardware) para que funcionem na 'velocidade do enlace'. Isso significa que eles podem processar pacotes tão rapidamente quanto os pacotes podem chegar ao enlace. Segundo, o desempenho relevante é aquele que as aplicações obtêm. Não é a capacidade do enlace, mas a vazão e o atraso após o processamento da rede e do transporte.

A redução dos overheads de software melhora o desempenho aumentando a vazão e diminuindo o atraso. Ela também pode reduzir a energia que é gasta na rede, que é uma consideração importante para computadores móveis. A maioria dessas ideias é do conhecimento dos projetistas de redes há anos e tem sido passada de geração a geração. Elas foram especificadas pela primeira vez por Mogul (1993); nosso tratamento é paralelo ao dele. Outra fonte relevante é Metcalfe (1993).

A velocidade do host é mais importante que a velocidade da rede

Longos anos de experiência mostraram que em quase todas as redes o sistema operacional e o overhead do protocolo dominam o tempo real no enlace. Por exemplo, teoricamente o tempo mínimo RPC em uma rede Ethernet a 1 Gbps é 102 μs, o que corresponde a uma solicitação mínima (512 bytes) seguida por uma resposta mínima (512 bytes). Na prática, superar o tempo de overhead do software e fazer o tempo da RPC ficar próximo ao esperado é um aperfeiçoamento substancial. Isso raramente acontece na prática.

Da mesma forma, o maior problema de trabalhar a 1 Gbps é conseguir que os bits passem do buffer do usuário para o enlace de fibra-óptica com rapidez suficiente, e fazer o host receptor processá-los à mesma velocidade com que eles chegam. Se duplicar a velocidade do host, com frequência você chegará perto de duplicar a vazão. Duplicar a capacidade da rede não tem nenhum efeito prático, pois em geral o gargalo está nos hosts.

Reduza o número de pacotes para reduzir o overhead

Cada segmento tem uma certa quantidade de overhead (por exemplo, o cabeçalho), assim como os dados (por exemplo, a carga útil). A largura de banda é exigida para os dois componentes. O processamento também é exigido para os dois componentes (por exemplo, o processamento do cabeçalho e a realização do checksum). Quando 1 milhão de bytes estão sendo enviados, o custo dos dados é o mesmo, não importa qual seja o tamanho do segmento. Porém, usar segmentos de 128 bytes significa 32 vezes o overhead por segmento ao usar segmentos de 4 KB. Os overheads de largura de banda e processamento se acumulam rapidamente para reduzir a vazão.

O overhead por pacote nas camadas inferiores amplifica esse efeito. Cada pacote recebido causa uma nova interrupção se o host estiver acompanhando. Em um processador moderno com pipeline, cada interrupção desfaz o pipeline com a CPU, interfere no cache, exige uma mudança no contexto de gerenciamento de memória, anula a tabela de previsão de desvio e força a gravação de um número substancial de registradores de CPU. Uma redução de n vezes no número de segmentos transmitidos causa uma redução de n vezes no overhead dos pacotes e nas interrupções.

Você poderá dizer que pessoas e computadores são fracos em multitarefas. Essa observação está por trás do desejo de enviar pacotes de MTU tão grandes que passarão pelo caminho da rede sem fragmentação. Mecanismos como o algoritmo de Nagle e a solução de Clark também são tentativas de evitar o envio de pacotes pequenos.

Minimize a movimentação de dados

O modo mais simples de implementar uma pilha de protocolos em camadas é com um módulo para cada camada. Infelizmente, isso ocasiona a cópia (ou, pelo menos, o acesso aos dados em múltiplas passadas), pois cada camada faz seu próprio trabalho. Por exemplo, após um pacote ser recebido pela NIC, ele normalmente é copiado para um buffer do núcleo. A partir daí, ele é copiado para um buffer da camada de rede para processamento pela camada de rede, depois para um buffer da camada de transporte para processamento pela camada de transporte, e finalmente para o processo da aplicação receptora. Não é raro que um pacote que chega seja copiado três ou quatro vezes antes que o segmento encapsulado nele seja entregue.

Toda essa operação de cópia pode diminuir bastante o desempenho, pois sistemas de memória são uma ordem de grandeza mais lenta do que as instruções registrador-registrador. Por exemplo, se 20 por cento das instruções realmente vão para a memória (ou seja, são perdas de cache), o que é provável quando se movimentam os pacotes que chegam, o tempo de execução médio da instrução é diminuído por um fator de 2,8 ($0{,}8 \times 1 + 0{,}2 \times 10$). O auxílio do hardware não ajudará aqui. O problema é, em grande parte, de cópia pelo sistema operacional.

Um sistema operacional inteligente reduzirá a cópia combinando o processamento de múltiplas camadas. Por exemplo, TCP e IP normalmente são implementados juntos (como 'TCP/IP'), de modo que não é necessário copiar a carga útil do pacote quando o processamento passa da camada de rede para a de transporte. Outro truque comum é realizar várias operações dentro de uma camada em uma única passada pelos dados. Por exemplo, os checksums normalmente são calculados enquanto se copiam os dados (quando eles precisam ser copiados) e o checksum recém-calculado é anexado ao final.

Minimize as mudanças de contexto

Uma regra relacionada é que as mudanças de contexto (por exemplo, do modo de núcleo para o modo de usuário) são fatais. Elas têm as mesmas propriedades ruins que as interrupções e cópia combinadas. Esse custo é o motivo pelo qual os protocolos de transporte normalmente são im-

plementados no núcleo. Assim como a redução da quantidade de pacotes, é possível reduzir as mudanças de contexto por meio de um procedimento de biblioteca que envia os dados para um buffer interno até que haja um volume substancial de dados. Da mesma forma, na estação receptora, os pequenos segmentos recebidos devem ser agrupados e repassados de uma só vez ao usuário, em vez de ser repassados individualmente, com a finalidade de minimizar as mudanças de contexto.

Na melhor das hipóteses, um pacote recebido provoca uma mudança de contexto do usuário atual para o núcleo, e depois uma mudança para o processo receptor, a fim de enviar os dados recém-chegados. No entanto, infelizmente, em muitos sistemas operacionais acontecem outras mudanças de contexto. Por exemplo, se o gerenciador da rede executar um processo especial no espaço do usuário, a chegada de um pacote provavelmente causará uma mudança de contexto do usuário atual para o núcleo, depois outra mudança do núcleo para o gerenciador de rede, seguida por outra de volta ao núcleo, e uma última do núcleo para o processo receptor. Essa sequência é ilustrada na Figura 6.44. Todas essas mudanças de contexto em cada pacote desperdiçam tempo de CPU e terão um efeito devastador sobre o desempenho da rede.

Prevenir o congestionamento é melhor do que remediá-lo

A velha máxima de que é melhor prevenir do que remediar se aplica ao congestionamento da rede. Quando uma rede está congestionada, pacotes são perdidos, há desperdício de largura de banda, atrasos inúteis acontecem, e muito mais. Remediar essa situação toma tempo e paciência. É muito melhor evitar que tudo isso aconteça. Impedir o congestionamento é como ser vacinado: dói um pouco na hora, mas evita algo que poderia causar muito mais dor no futuro.

Evite os timeouts

Os timers são necessários nas redes, mas devem ser usados com moderação, e os timeouts devem ser minimizados. Quando um timer expira, em geral alguma ação é repetida. Se essa repetição for imprescindível, tudo bem, mas repeti-la sem necessidade é um desperdício.

O melhor modo de evitar trabalho desnecessário é cuidar para que os timers sejam programados de maneira um pouco conservadora. Um timer que demora tempo demais para expirar aumenta o atraso de uma conexão, caso haja a (improvável) perda de um segmento. Um timer que expira cedo demais utiliza muitos recursos do host, desperdiça largura de banda e aumenta a carga em dezenas de roteadores sem uma boa razão para tal.

6.6.4 Processamento rápido de segmentos

Agora que já vimos as regras gerais, vamos examinar alguns métodos específicos para tornar esse processamento de segmento mais rápido. Para obter mais informações, consulte Clark et al. (1989) e Chase et al. (2001).

O overhead do processamento de segmentos tem dois componentes: o overhead por segmento e o overhead por byte. Ambos devem ser combatidos. O segredo para o processamento rápido de segmentos é separar a situação normal (a transferência de dados em um sentido) e dar a ela um tratamento especial. Muitos protocolos tendem a enfatizar o que fazer quando algo sai errado (por exemplo, um pacote perdido), mas, para que os protocolos funcionem rapidamente, o projetista deve procurar reduzir o tempo de processamento quando tudo correr bem. Minimizar o tempo de processamento quando ocorre um erro é secundário.

Ainda que seja necessária uma sequência especial de segmentos para chegar ao estado *ESTABLISHED*, uma vez nele, o processamento de segmentos se dá de forma simples até um dos lados começar a fechar a conexão. Vamos começar examinando o lado transmissor no estado *ESTABLISHED* quando há dados a ser transmitidos. Para facilitar a compreensão, faremos a suposição de que a entidade de transporte está no núcleo, embora as mesmas ideias também sejam verdadeiras caso ela seja uma biblioteca ou um processo do espaço do usuário dentro do processo transmissor. Na Figura 6.45, o processo transmissor fica preso no núcleo para executar um *SEND*. A primeira providência da entidade de transporte é realizar um teste para verificar se esse é o caso normal: o estado é *ESTABLISHED*, nenhum dos lados está tentando fechar a conexão, um segmento regular (e não fora da banda) completo está sendo transmitido e há espaço

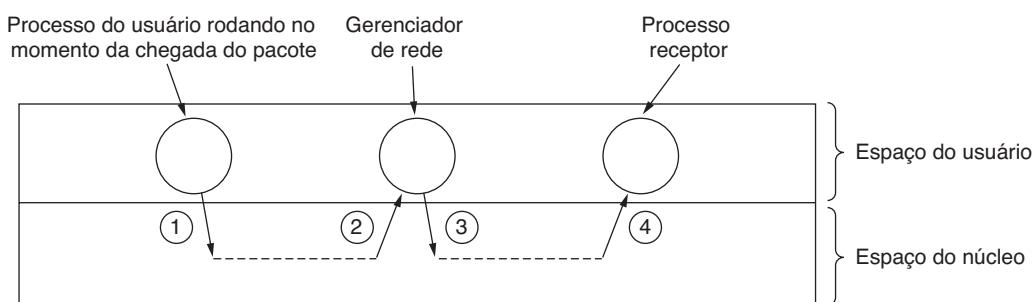



Figura 6.44 | Quatro mudanças de contexto para manipular um pacote por meio de um gerenciador de rede do espaço do usuário.

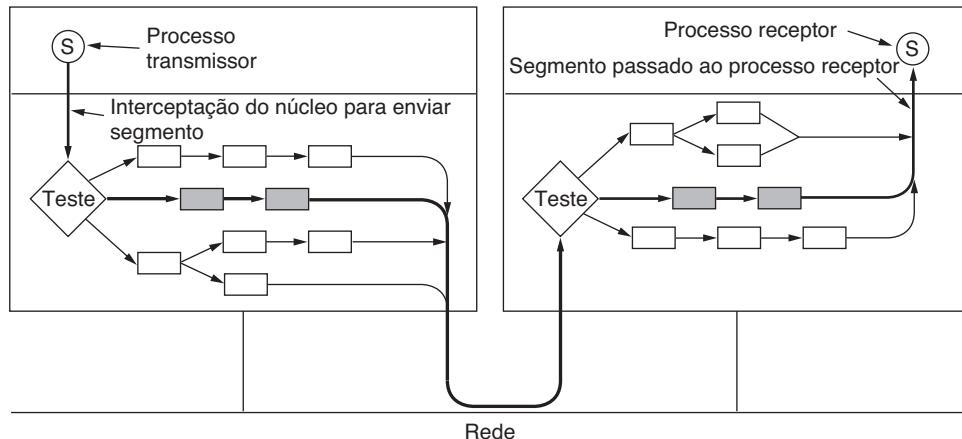



Figura 6.45 | O caminho rápido do transmissor para o receptor é ilustrado pela linha mais grossa. As etapas do processamento nesse caminho estão sombreadas.

de janela suficiente no receptor. Se todas as condições forem atendidas, não serão necessários outros testes e será possível seguir o caminho mais rápido pela entidade de transporte transmissora. Em geral, esse caminho é seguido durante a maior parte do tempo.

No caso normal, os cabeçalhos dos segmentos de dados consecutivos são quase idênticos. Para tirar proveito desse fato, um protótipo de cabeçalho é armazenado na entidade de transporte. No início do caminho rápido, ele é copiado com a maior velocidade possível para um buffer auxiliar, palavra por palavra. Esses campos que mudam de segmento para segmento são então sobregravados no buffer. Com frequência, esses campos são facilmente derivados de variáveis de estado, como o próximo número de sequência. Em seguida, um ponteiro para o cabeçalho do segmento completo e um ponteiro para os dados do usuário são repassados à camada de rede. Aqui, pode-se utilizar a mesma estratégia (não mostrada na Figura 6.45). Por último, a camada de rede repassa o pacote resultante à camada de enlace de dados para transmissão.

Como um exemplo do funcionamento desse princípio na prática, consideraremos o TCP/IP. A Figura 6.46(a) mostra o cabeçalho do TCP. Os campos iguais entre segmentos consecutivos em um fluxo de uma só via são mostrados sombreados. Tudo o que a entidade de transporte transmissora tem de fazer é copiar as cinco palavras do protótipo do cabeçalho para o buffer de saída, preencher o próximo número de sequência (copiando-o de uma palavra na memória), calcular o checksum e incrementar o número de sequência na memória. Em seguida, a entidade pode passar o cabeçalho e os dados para um procedimento especial do IP, a fim de transmitir um segmento máximo regular. Depois disso, o IP copia seu protótipo de cabeçalho de cinco palavras [ver Figura 6.46(b)] para o buffer, preenche o campo *Identificação* e calcula seu checksum. Com isso, o pacote fica pronto para transmissão.

Agora, vamos ver o processamento do caminho rápido no lado receptor da Figura 6.45. A primeira etapa é localizar o registro de conexão para o segmento recebido. No caso do TCP, o registro de conexão pode ser armazenado em uma tabela de hash cuja chave é alguma função simples dos dois endereços IP e das duas portas. Uma vez que o registro de conexão é localizado, ambos os endereços e ambas as portas devem ser comparados, a fim de verificar se o registro correto foi encontrado.

Figura 6.46 | (a) O cabeçalho TCP. (b) O cabeçalho IP. Em ambos os casos, eles foram tirados do protótipo sem nenhuma alteração.

Uma otimização que frequentemente acelera ainda mais a pesquisa de registros de conexões deve manter um ponteiro para a última conexão usada e experimentar esse registro primeiro. Clark et al. (1989) utilizaram essa opção e observaram uma taxa de aproveitamento de mais de 90 por cento.

Em seguida, o segmento é analisado para verificar se é um segmento normal, ou seja, se o estado é *ESTABLISHED*, se nenhum dos lados está tentando encerrar a conexão, se o segmento está completo e não tem flags especiais definidos, e se o número de sequência é o esperado. A realização desses testes exige apenas algumas instruções. Se todas as condições forem atendidas, esse procedimento será chamado de caminho rápido especial do TCP.

O caminho rápido atualiza o registro da conexão e copia os dados para o usuário. Enquanto copia, ele calcula o checksum, eliminando assim uma passagem extra pelos dados. Se o checksum estiver correto, o registro da conexão será atualizado e uma confirmação será enviada de volta. O esquema geral primeiramente é fazer uma verificação rápida para ver se o cabeçalho é o que se espera, e depois fazer um procedimento especial para tratar esse caso, chamado **prognóstico de cabeçalho**. Muitas implementações TCP o utilizam. Quando essa otimização e todas as outras descritas neste capítulo são aplicadas ao mesmo tempo, é possível fazer o TCP funcionar a 90 por cento da velocidade de uma cópia local memória a memória, supondo-se que a rede seja suficientemente rápida.

Duas outras áreas nas quais pode haver ganhos importantes de desempenho são o gerenciamento de buffers e o gerenciamento de timers. A grande questão no gerenciamento de buffers é evitar cópias desnecessárias, como mencionamos antes. O gerenciamento de timers é importante porque quase todos os timers programados não chegam a expirar. Eles são programados para prevenir a perda de segmentos, mas a maioria dos segmentos chega de forma correta, assim como suas confirmações. Consequentemente, é importante otimizar o gerenciamento, no caso de timers que raras vezes expiram.

Um esquema comum é usar uma lista encadeada contendo os eventos de timers ordenados por tempo de expiração. A entrada do cabeçalho contém um contador que informa quantos pulsos faltam para a expiração. Cada entrada sucessiva contém um contador que informa quantos pulsos a separam da anterior. Assim, se os timers expirarem em 3, 10 e 12 pulsos, os contadores serão 3, 7 e 2, respectivamente.

A cada pulso de clock, o contador na entrada do cabeçalho é decrementado. Quando ele chega a zero, seu evento é processado e o próximo item da lista torna-se o cabeçalho. Seu contador não precisa ser alterado. Nesse esquema, a inclusão e a exclusão de timers são operações de alto custo, com tempos de execução proporcionais ao tamanho da lista.

Uma estratégia muito mais eficiente pode ser usada se o intervalo máximo do timer for fixo e previamente conhecido. Aqui pode ser usada uma array chamada **roda de sincronismo**, como mostra a Figura 6.47. Cada slot corresponde a um pulso de clock. O tempo atual é $T = 4$. Os timers foram programados para expirar a 3, 10 e 12 pulsos a partir de agora. Se, de repente, um novo timer for programado para expirar em 7 pulsos, outra entrada será criada no slot 11. Da mesma forma, se o timer programado para $T + 10$ tiver de ser cancelado, a lista que começa no slot 14 deverá ser verificada e a entrada correspondente removida. Observe que a array da Figura 6.47 não contém nenhum timer acima de $T + 15$.

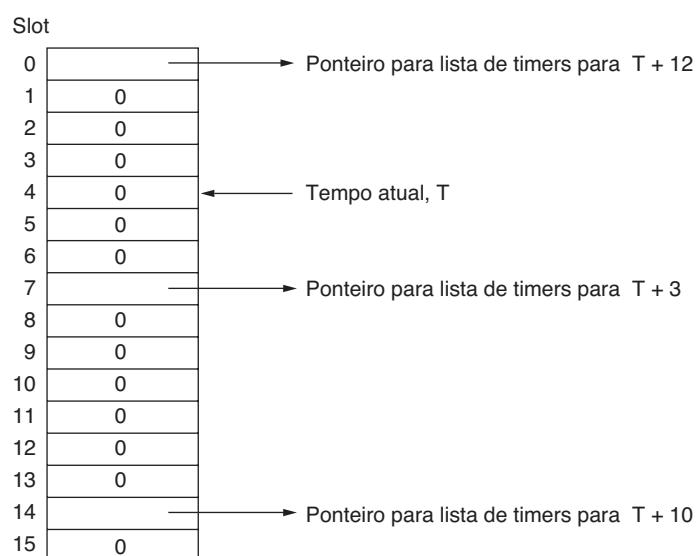



Figura 6.47 | Uma roda de sincronismo.

Quando há um pulso de clock, o ponteiro atual é avançado um slot (de forma circular). Se a entrada indicada for diferente de zero, todos os seus timers serão processados. Muitas variações sobre essa ideia básica são descritas em Varghese e Lauck (1987).

6.6.5 Compactação de cabeçalho

Estivemos examinando redes rápidas por muito tempo. Há muito mais. Agora, vamos considerar o desempenho nas redes sem fios e em outras redes em que a largura de banda é limitada. Reduzir o overhead de software pode ajudar computadores móveis a trabalhar com mais eficiência, mas não melhora o desempenho quando os enlaces da rede são o gargalo.

Para usar bem a largura de banda, os cabeçalhos de protocolos e as cargas úteis devem ser transportados com o mínimo de bits. Para as cargas úteis, isso significa usar codificações compactas de informação, como imagens que estão em formato JPEG em vez de um mapa de bits, ou formatos de documento como PDF, que incluem compactação. Isso também significa mecanismos de caching em nível de aplicação, como caches Web que reduzem transferências em primeiro lugar.

E os cabeçalhos de protocolo? Na camada de enlace, os cabeçalhos para redes sem fios normalmente compactam porque foram projetados visando a uma largura de banda escassa. Por exemplo, cabeçalhos 802.16 possuem identificadores de conexão curtos, ao invés de endereços mais longos. Porém, protocolos de camada mais alta, como IP, TCP e UDP, vêm em uma versão para todas as camadas de enlace, e eles não são projetados com cabeçalhos compactos. De fato, o processamento rápido para reduzir o overhead de software normalmente leva a cabeçalhos que não são tão compactos quanto poderiam ser de outra forma (por exemplo, o IPv6 tem cabeçalhos mais compactados do que o IPv4).

Os cabeçalhos de camada mais alta podem gerar um ganho de desempenho significativo. Considere, por exemplo, os dados VoIP que são transportados com a combinação de IP, UDP e RTP. Esses protocolos exigem 40 bytes de cabeçalho (20 para IPv4, 8 para UDP e 12 para RTP). Com IPv6, a situação é ainda pior: 60 bytes, incluindo 40 bytes de cabeçalho IPv6. Os cabeçalhos podem acabar sendo a maior parte dos dados transmitidos, consumindo mais da metade da largura de banda.

A **compactação de cabeçalho** é usada para reduzir a largura de banda ocupada pelos cabeçalhos dos protocolos das camadas superiores nos enlaces. Esquemas projetados especialmente são usados no lugar dos métodos de uso geral. Isso porque os cabeçalhos são curtos, de modo que não são muito bem compactados individualmente, e a descompactação exige que todos os dados anteriores sejam recebidos. Isso não acontecerá se um pacote for perdido.

A compactação de cabeçalho obtém grandes ganhos usando o conhecimento do formato do protocolo. Um dos primeiros esquemas foi projetado por Van Jacobson (1990) para compactação de cabeçalhos TCP/IP por enlaces seriais lentos. Ele é capaz de compactar um cabeçalho TCP/IP típico de 40 bytes para uma média de 3 bytes. O truque para esse método é indicado na Figura 6.46. Muitos dos campos de cabeçalho não mudam de um pacote para outro. Por exemplo, não é preciso enviar o mesmo TTL IP ou os mesmos números de porta TCP em todo e qualquer pacote. Eles podem ser omitidos no lado transmissor do enlace e preenchidos no lado receptor.

De modo semelhante, outros campos mudam de uma maneira previsível. Por exemplo, com exceção da perda, o número de sequência do TCP avança com os dados. Nesses casos, o receptor pode prever o valor provável. O número real só precisa ser transportado quando for diferente do que é esperado. Mesmo assim, ele pode ser transportado como uma pequena mudança do valor anterior, como quando o número de confirmação aumenta quando novos dados são recebidos no sentido oposto.

Com a compactação de cabeçalho, é possível ter cabeçalhos simples em protocolos de camada superior e codificações compactas para enlaces com pouca largura de banda. **ROHC** (**RObust Header Compression**) é uma versão moderna da compactação de cabeçalho que é definida como uma estrutura na RFC 5795. Ela é projetada para tolerar a perda que pode ocorrer nos enlaces sem fios. Existe um perfil para cada conjunto de protocolos a ser compactado, como IP/UDP/RTP. Os cabeçalhos compactados são transportados referindo-se a um contexto, que é basicamente uma conexão; os campos de cabeçalho podem ser facilmente previstos para pacotes da mesma conexão, mas não para pacotes de conexões diferentes. Na operação típica, ROHC reduz cabeçalhos IP/UDP/RTP de 40 bytes para 1 a 3 bytes.

Embora a compactação de cabeçalho vise principalmente a reduzir as necessidades de largura de banda, ela também pode ser útil para reduzir o atraso. O atraso é composto de atraso de propagação, que é fixo para um caminho na rede, e atraso de transmissão, que depende da largura de banda e da quantidade de dados a ser enviados. Por exemplo, um enlace de 1 Mbps envia 1 bit em 1 μs. No caso de mídia por redes sem fios, a rede é relativamente lenta, de modo que o atraso de transmissão pode ser um fator importante no atraso geral e um atraso consistentemente baixo é importante para a qualidade do serviço.

A compactação de cabeçalho pode ajudar reduzindo a quantidade de dados enviados e, portanto, reduzindo o atraso na transmissão. O mesmo efeito pode ser obtido enviando-se pacotes menores. Isso compensará um maior overhead de software por um menor atraso de transmissão. Observe que outra fonte de atraso em potencial é o atraso no enfileiramento para acessar o enlace sem fios. Isso também pode ser significativo, pois os enlaces sem fios são muito utilizados como recurso limitado em uma rede. Nesse caso, ele precisa ter mecanismos de qualidade de serviço que dão um atraso baixo aos pacotes em tempo real. Apenas a compactação de cabeçalho não é suficiente.

6.6.6 Protocolos para redes longas de banda larga

Desde a década de 90, tem havido redes de gigabits que transmitem dados a grandes distâncias. Devido à combinação de uma rede rápida de banda larga 'fat networks' e de longo atraso, essas redes são chamadas **redes longas de banda larga**. Quando essas redes surgiram, a primeira reação das pessoas foi usar os protocolos existentes nelas, mas diversos problemas apareceram rapidamente. Nesta seção, discutiremos alguns dos problemas com o aumento da velocidade e o atraso dos protocolos de rede.

O primeiro problema é que muitos protocolos utilizam números de sequência de 32 bits. Quando a Internet começou, as linhas entre os roteadores eram principalmente linhas concedidas de 45 kbps, de modo que um host transmitindo em velocidade plena levava mais de uma semana para percorrer todos os números de sequência. Para os projetistas do TCP, 2^{32} era uma aproximação do infinito, pois havia pouco perigo de pacotes antigos ficarem rodando uma semana depois que fossem transmitidos. Com a Ethernet de 10 Mbps, esse tempo passou para 57 minutos, muito menor, mas ainda assim manejável. Com uma Ethernet de 1 Gbps jogando dados na Internet, o tempo é cerca de 34 segundos, muito abaixo do tempo de vida máximo do pacote de 120 segundos na Internet. De repente, 2^{32} não é uma aproximação tão boa do infinito, pois um transmissor rápido pode percorrer o espaço de sequência enquanto pacotes antigos ainda existem na rede.

O problema é que muitos projetistas de protocolo simplesmente consideraram, sem afirmar isso, que o tempo exigido para ocupar o espaço de sequência inteiro seria muito maior que o tempo de vida máximo do pacote. Consequentemente, não havia necessidade sequer de se preocupar com o problema de duplicatas antigas ainda existindo quando os números de sequência fossem reiniciados. Em velocidades de gigabits, essa suposição não declarada falha. Felizmente, é possível estender o número de sequência efetivo tratando o período de tempo que pode ser transportado como uma opção no cabeçalho TCP de cada pacote como os bits de alta ordem. Esse mecanismo é chamado PAWS (Protection Against Wrapped Sequence numbers) e é descrito na RFC 1323.

O segundo problema é que o tamanho da janela de controle de fluxo precisa ser bastante aumentado. Por exemplo, considere o envio de uma rajada de 64 KB de dados de San Diego para Boston a fim de preencher o buffer de 64 KB do receptor. Suponha que o enlace seja de 1 Gbps e o atraso unidirecional da velocidade da luz na fibra seja de 20 ms. Inicialmente, a $t = 0$, o canal está vazio, conforme ilustra a Figura 6.48(a). Somente 500 μs depois, na Figura 6.48(b), todos os segmentos estão na fibra. O segmento inicial agora estará em algum ponto nas proximidades de Brawley, ainda ao sul da Califórnia. Porém, o transmissor precisa parar até que ele receba uma atualização de janela.

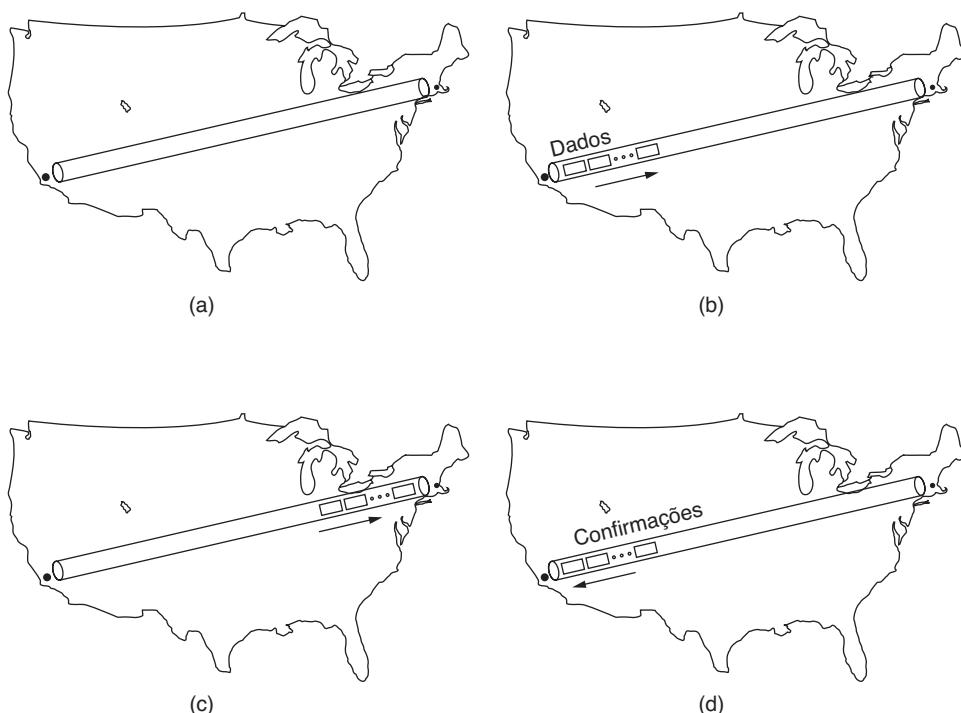



Figura 6.48 | O estado da transmissão de 1 Mbit de San Diego para Boston. (a) Em $t = 0$. (b) Após 500 μs. (c) Após 20 ms. (d) Após 40 ms.

Após 20 ms, o segmento inicial alcança Boston, como mostra a Figura 6.48(c), e é confirmado. Finalmente, 40 ms depois de iniciar, a primeira confirmação retorna ao transmissor e a segunda rajada pode ser transmitida. Como a linha de transmissão foi usada por 1,25 ms dos 100, a eficiência é de cerca de 1,25 por cento. Essa situação é típica de um protocolo mais antigo rodando por linhas de gigabits.

Uma quantidade útil para ter em mente ao analisar o desempenho da rede é o **produto largura de banda-atraso**. Ele é obtido multiplicando-se a largura de banda (em bits/s) pelo tempo de atraso de ida e volta (em segundos). O produto é a capacidade do canal do transmissor ao receptor e de volta (em bits).

Para o exemplo da Figura 6.48, o produto largura de banda-atraso é de 40 milhões de bits. Em outras palavras, o transmissor teria que transmitir uma rajada de 40 milhões de bits para poder continuar em velocidade plena até que a primeira confirmação retornasse. É necessário que haja esse número de bits para encher o canal (nas duas direções). É por isso que uma rajada de meio milhão de bits só atinge uma eficiência de 1,25 por cento: isso significa apenas 1,25 por cento da capacidade do canal.

A conclusão que podemos tirar aqui é que, para obter um bom desempenho, a janela do receptor precisa ser pelo menos tão grande quanto o produto largura de banda-atraso, e de preferência um pouco maior, pois o receptor pode não responder instantaneamente. Para uma linha de gigabit transcontinental, pelo menos 5 MB são necessários.

O terceiro problema, também relacionado, é que esquemas de retransmissão simples, como o protocolo go-back-n, não funcionam bem em linhas com um produto largura de banda-atraso grande. Considere o enlace transcontinental de 1 Gbps com um tempo de transmissão de ida e volta de 40 ms. Um transmissor pode enviar 5 MB em uma viagem de ida e volta. Se um erro for detectado, isso será 40 ms antes que o transmissor seja informado a respeito. Se o protocolo go-back-n for usado, o transmissor terá que retransmitir não apenas o pacote com problema, mas também os 5 MB de pacotes que vieram depois. É claro que isso é um grande desperdício de recursos. Protocolos mais complexos, como a repetição seletiva, são necessários.

O quarto é que as linhas de gigabits são fundamentalmente diferentes das linhas de megabits, pois as longas linhas de gigabits são limitadas por atraso em vez de limitadas por largura de banda. Na Figura 6.49, mostramos o tempo gasto para transferir um arquivo de 1 Mbit por 4.000 km em diversas velocidades de transmissão. Em velocidades de até 1 Mbps, o tempo de transmissão é dominado pela taxa em que os bits podem ser enviados. Por volta de 1 Gbps, o atraso de ida e volta de 40 ms é superior ao tempo de 1 ms gasto para colocar os bits na fibra. Outros aumentos na largura de banda dificilmente terão algum efeito.

A Figura 6.49 tem implicações infelizes para os protocolos de rede. Ela diz que os protocolos stop-and-wait, como RPC, têm um limite superior inerente em seu desempenho. Esse limite é ditado pela velocidade da luz. Nenhuma quantidade de progresso tecnológico na óptica conseguirá melhorar as coisas (porém, novas leis da física ajudariam). A menos que possa ser encontrado algum outro uso para uma linha de gigabits enquanto um host está esperando uma resposta, a linha de gigabits não é melhor do que uma linha de megabits, apenas mais cara.

Um quinto problema é que as velocidades de comunicação têm melhorado com mais rapidez que as velocidades de computação. (Nota para os engenheiros de computação: saiam e vençam os engenheiros da comunicação! Estamos contando com vocês.) Na década de 70, a ARPANET funcionava a 56 kbps e tinha computadores que funcionavam

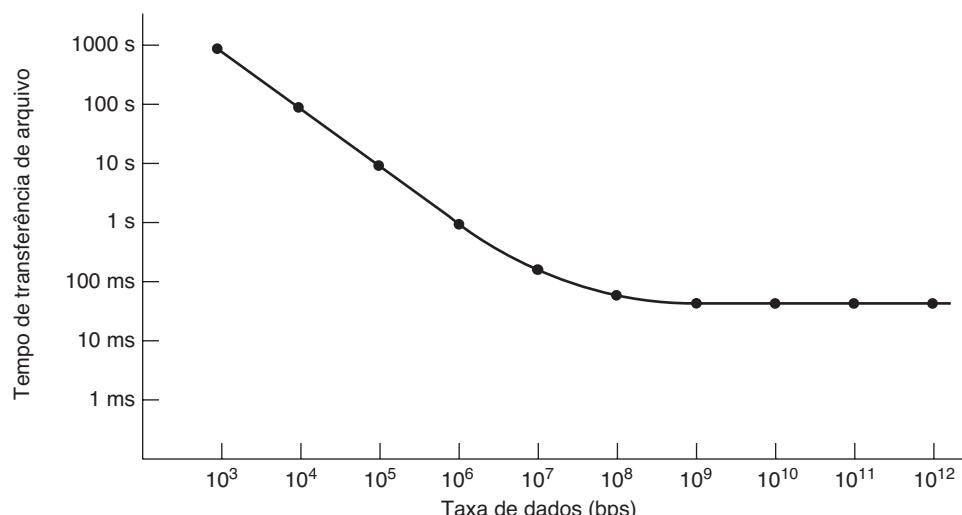



Figura 6.49 | Tempo para transferir e confirmar um arquivo de 1 Mbit por uma linha de 4.000 km.

em aproximadamente 1 MIPS. Compare esses números com computadores de 1.000 MIPS trocando pacotes por uma linha de 1 Gbps. O número de instruções por byte diminuiu por um fator de mais de 10. Os números exatos são discutíveis, dependendo das datas e cenários, mas a conclusão é esta: há menos tempo disponível para processamento de protocolo do que havia antes, de modo que os protocolos se tornaram mais simples.

Agora, vamos passar dos problemas para as maneiras de lidar com eles. O princípio básico que todos os projetistas de redes de alta velocidade precisam aprender de cor é:

Projete visando à velocidade, e não à otimização da largura de banda.

Os protocolos antigos normalmente eram projetados para minimizar o número de bits nos enlaces, geralmente usando campos pequenos e compactando-os em bytes e palavras. Essa preocupação ainda é válida para redes sem fios, mas não para redes de gigabits. O processamento de protocolo é o problema, de modo que os protocolos precisam ser projetados para minimizá-lo. Os projetistas do IPv6 certamente entenderam esse princípio.

Um modo atraente de aumentar a velocidade é criar interfaces de rede rápidas em hardware. A dificuldade com essa estratégia é que, a menos que o protocolo seja incrivelmente simples, o hardware simplesmente significa uma placa com uma segunda CPU e seu próprio programa. Para garantir que o coprocessador de rede seja mais barato que a CPU principal, ele normalmente é um chip mais lento. A consequência desse projeto é que grande parte do tempo na CPU principal (rápida) fica ocioso aguardando que a segunda CPU (lenta) realize o trabalho crítico. É um mito pensar que a CPU principal tem outro trabalho a fazer enquanto espera. Além do mais, quando duas CPUs de uso geral se comunicam, pode haver condições de *race*, de modo que protocolos complicados são necessários entre os dois processadores para sincronizá-los corretamente e evitar *races*. Normalmente, a melhor técnica é tornar os protocolos simples e deixar que a CPU principal faça o trabalho.

O layout de pacotes é uma consideração importante nas redes de gigabits. O cabeçalho deve conter o mínimo possível de campos, a fim de reduzir o tempo de processamento. Esses campos devem ser grandes o suficiente para realizar o trabalho e ter alinhamento de palavras com a finalidade de facilitar o processamento. Nesse contexto, 'grande o suficiente' significa que não ocorrerão problemas como repetição de números de sequência enquanto ainda existem pacotes antigos, receptores incapazes de anunciar espaço de janela suficiente porque o campo da janela é muito pequeno e assim por diante.

O tamanho máximo dos dados deve ser grande, para reduzir o overhead de software e permitir uma operação eficiente. Para redes de alta velocidade, 1.500 bytes é muito pouco, motivo pelo qual a Ethernet de gigabit admite quadros jumbo de até 9 KB e o IPv6 admite pacotes jumbograma com mais de 64 KB.

Agora, vamos examinar a questão do feedback nos protocolos de alta velocidade. Devido ao loop de atraso (relativamente) longo, o feedback deverá ser evitado: o receptor gasta muito tempo para sinalizar o transmissor. Um exemplo de feedback é controlar a taxa de transmissão usando um protocolo de janela de deslizante. Os protocolos do futuro poderão passar para protocolos baseados em taxa, para evitar os (longos) atrasos inerentes ao envio de atualizações de janela do receptor ao transmissor. Nesse protocolo, o transmissor pode enviar tudo o que desejar, desde que não envie mais rápido do que alguma taxa que o transmissor e o receptor combinaram antecipadamente.

Um segundo exemplo de feedback é o algoritmo de partida lenta de Jacobson. Esse algoritmo promove várias sondagens para verificar quanto a rede pode manipular. Em uma rede de alta velocidade, fazer meia dúzia de pequenas sondagens para ver como a rede se comporta desperdiça um grande volume de largura de banda. Um esquema mais eficiente é fazer com que o transmissor, o receptor e a rede reservem os recursos necessários no momento de estabelecer a conexão. Reservar recursos antecipadamente também oferece a vantagem de tornar mais fácil reduzir o jitter. Em resumo, buscar altas velocidades leva o projeto inexoravelmente em direção a uma operação orientada a conexões, ou algo bem parecido.

Outro recurso valioso é a possibilidade de enviar um volume normal de dados junto com a solicitação de conexão. Desse modo, é possível economizar o tempo de uma viagem de ida e volta.

6.7 Redes tolerantes a atrasos

Vamos terminar este capítulo descrevendo um novo tipo de transporte que um dia poderá ser um componente importante da Internet. O TCP e a maioria dos outros protocolos de transporte são baseados na hipótese de que o transmissor e o receptor estão continuamente conectados por algum caminho funcional, ou então o protocolo falha e os dados não podem ser entregues. Em algumas redes, normalmente não existe um caminho fim a fim. Um exemplo é uma rede espacial, como satélites LEO (Low-Earth Orbit) entrando e saindo do alcance das estações terrestres. Determinado satélite pode ser capaz de se comunicar com uma estação terrestre somente em certos momentos, e dois satélites podem nunca ser capazes de se comunicar um com o outro em momento algum, mesmo por meio da estação terrestre, pois um dos satélites pode sempre estar fora de alcance. Mais exemplos de redes envolvem submarinos, ônibus, telefones móveis e outros dispositivos com computadores para os quais existe conectividade intermitente, devido à mobilidade ou a condições extremas.

Nessas redes ocasionalmente conectadas, os dados ainda podem ser comunicados armazenando-os em nós e encaminhando-os mais adiante, quando existe um enlace funcional. Essa técnica é chamada **comutação de mensagens**. Por fim, os dados serão repassados ao destino. Uma rede cuja arquitetura é baseada nessa técnica é chamada rede tolerante a atrasos, ou **DTN** (**Delay-Tolerant Network**, ou **Disruption-Tolerant Network**).

O trabalho sobre DTNs começou em 2002, quando a IETF montou um grupo de pesquisa sobre o assunto. A inspiração para as DTNs veio de uma fonte improvável: esforços para enviar pacotes no espaço. As redes espaciais precisam lidar com comunicação intermitente e atrasos muito longos. Kevin Fall observou que as ideias para essas Internets interplanetárias poderiam ser aplicadas às redes terrestres em que a conectividade intermitente era a norma (Fall, 2003). Esse modelo oferece uma generalização útil da Internet, em que o armazenamento e os atrasos podem ocorrer durante a comunicação. A entrega de dados é semelhante à entrega no sistema postal, ou correio eletrônico, em vez da comutação de pacotes nos roteadores.

Desde 2002, a arquitetura DTN tem sido refinada, e as aplicações do modelo DTN aumentaram. Como uma aplicação fundamental, considere grandes conjuntos de dados de muitos terabytes que são produzidos por experimentos científicos, eventos de mídia ou serviços baseados na Web e que precisam ser copiados para centros de dados em diferentes locais no mundo inteiro. Os operadores gostariam de enviar esse tráfego intenso em horários fora de pico, utilizando a largura de banda que já foi paga, mas que não está sendo usada, e podem tolerar algum atraso. Isso é como fazer os backups à noite, quando outras aplicações não estão utilizando a rede intensamente. O problema é que, para os serviços globais, os horários fora de pico são diferentes em vários locais no mundo. Pode haver pouca sobreposição nos horários em que os centros de dados em Boston e Perth possuem largura de banda de rede fora de pico, pois noite para uma cidade é dia para a outra.

Porém, os modelos de DTN permitem armazenamento e atrasos durante a transferência. Com esse modelo, é possível enviar o conjunto de dados de Boston a Amsterdã usando a largura de banda fora de pico, pois as cidades têm fusos horários afastados em apenas 6 horas. O conjunto de dados é então armazenado em Amsterdã até que haja largura de banda fora de pico entre Amsterdã e Perth. Em seguida, ele é enviado para Perth para concluir a transferência. Laoutaris et al. (2009) estudaram esse modelo e descobriram que ele pode oferecer capacidade substancial com pouco custo, e que o uso de um modelo DTN normalmente dobra a capacidade em comparação com o modelo fim a fim tradicional.

A seguir, vamos descrever a arquitetura DTN da IETF e seus protocolos.

6.7.1 Arquitetura DTN

A principal suposição na Internet que as DTNs buscam relativizar é que existe um caminho fim a fim entre uma origem e um destino por toda a duração de uma sessão de comunicação. Quando isso não acontece, os protocolos normais da Internet falham. As DTNs contornam a falta de conectividade fim a fim com uma arquitetura que é baseada na comutação de mensagens, como mostra a Figura 6.50. Ela também tem a finalidade de tolerar enlaces com pouca confiabilidade e grandes atrasos. A arquitetura é especificada na RFC 4838.

Na terminologia DTN, uma mensagem é chamada de **bundle**. Os nós DTN são equipados com armazenamento, normalmente armazenamento persistente, como um disco ou memória flash. Eles armazenam bundles até que os enlaces fiquem disponíveis e depois encaminham os bundles. Os enlaces funcionam de modo intermitente. A Figura 6.50

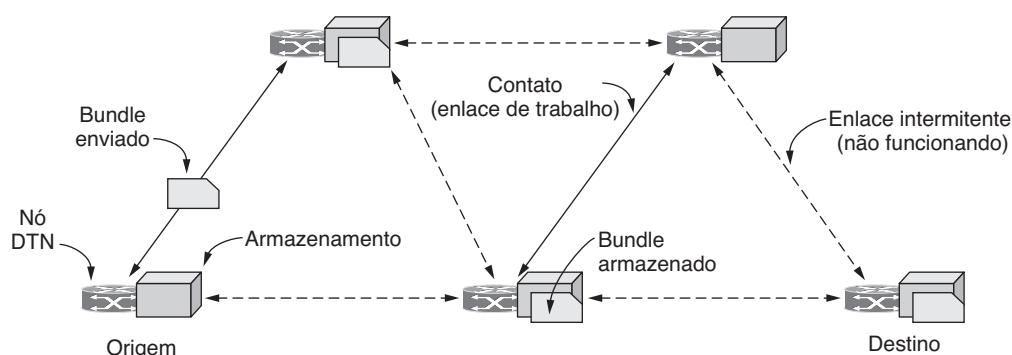



Figura 6.50 | Arquitetura das redes tolerantes a atrasos.

mostra cinco enlaces intermitentes que não estão funcionando atualmente, e dois enlaces que estão funcionando. Um enlace funcionando é chamado de **contato**. A Figura 6.50 também mostra bundles armazenados em dois nós DTN aguardando contatos para enviar os bundles adiante. Desse modo, os bundles são repassados por meio de contatos da origem para o seu destino.

O armazenamento e o encaminhamento de bundles em nós DTN parecem semelhantes ao enfileiramento e encaminhamento de pacotes nos roteadores, mas existem diferenças qualitativas. Nos roteadores da Internet, o enfileiramento ocorre por milissegundos ou, no máximo, segundos. Nos nós DTN, os bundles podem ser armazenados por horas, até que um ônibus chegue à cidade, enquanto um avião completa um voo, até que um nó de sensores colha energia solar suficiente para funcionar, até que um computador dormindo acorde e assim por diante. Esses exemplos também apontam para uma segunda diferença, ou seja, que os nós podem se mover (com um ônibus ou avião) enquanto mantêm dados armazenados, e esse movimento pode até mesmo ser uma parte essencial da remessa de dados. Os roteadores na Internet não têm permissão para se mover. O processo inteiro de mover bundles poderia ser mais conhecido como 'store-carry-forward'.

Como exemplo, considere o cenário mostrado na Figura 6.51, que foi o primeiro uso dos protocolos DTN no espaço (Wood et al., 2008). A origem dos bundles é um satélite LEO que está registrando imagens da Terra como parte da Constelação de Monitoramento de Desastres por satélites. As imagens devem ser retornadas ao ponto de coleta. Entretanto, o satélite tem contato apenas intermitente com três estações terrestres enquanto orbita a Terra. Ele entra em contato com cada estação terrestre, uma de cada vez. Cada um dentre os satélites, as estações terrestres e o ponto de coleta atuam como um nó DTN. Em cada contato, um bundle (ou uma parte dele) é enviado a uma estação terrestre. Os bundles são então enviados por uma rede terrestre de retorno para um ponto de coleta, para concluir a transferência.

A vantagem principal da arquitetura DTN nesse exemplo é que ela naturalmente se ajusta à situação do satélite precisando armazenar imagens, pois não existe conectividade no momento em que a imagem é tirada. Existem mais duas vantagens. Primeiro, pode não haver um único contato longo o suficiente para enviar as imagens, mas elas podem se espalhar pelos contatos com três estações terrestres. Segundo, o uso do enlace entre o satélite e a estação terrestre é desacoplado do enlace pela rede de retorno. Isso significa que o download do satélite não está limitado por um enlace terrestre lento. Ele pode prosseguir em velocidade completa, com o bundle armazenado na estação terrestre até que possa ser repassado para o ponto de coleta.

Uma questão importante que não é especificada pela arquitetura é como encontrar boas rotas por meio de nós DTN. Boas rotas dependem da natureza que a arquitetura descreve quando envia dados e também de quais são os contatos. Alguns contatos são conhecidos com antecedência. Um bom exemplo é o movimento de corpos celestes no espaço. Para o experimento espacial, eram conhecidos com antecedência o momento em os contatos ocorreriam, o fato que os intervalos de contato variavam de 5 a 14 minutos por passada em cada estação terrestre e que a capacidade do downlink era de 8,134 Mbps. Com esse conhecimento, o transporte de um bundle de imagens pode ser planejado com antecedência.

Em outros casos, os contatos podem ser previstos, mas com menos certeza. Alguns exemplos incluem ônibus que fazem contato uns com os outros de maneiras regulares, devido a um horário fixo, embora com alguma variação, e os tempos e a quantidade de largura de banda fora do pico nos protocolos de ISP, que são previstos a partir de dados do passado. No outro extremo, os contatos são ocasionais e aleatórios. Um exemplo é o transporte de dados de um usuário para outro em telefones móveis, dependendo de quais usuários fazem contato uns com os outros durante o dia. Quando há imprevisibilidade nos contatos, uma estratégia de roteamento é enviar cópias do bundle por diferentes caminhos, na esperança de que uma das cópias seja entregue ao destino antes que o tempo de vida seja alcançado.

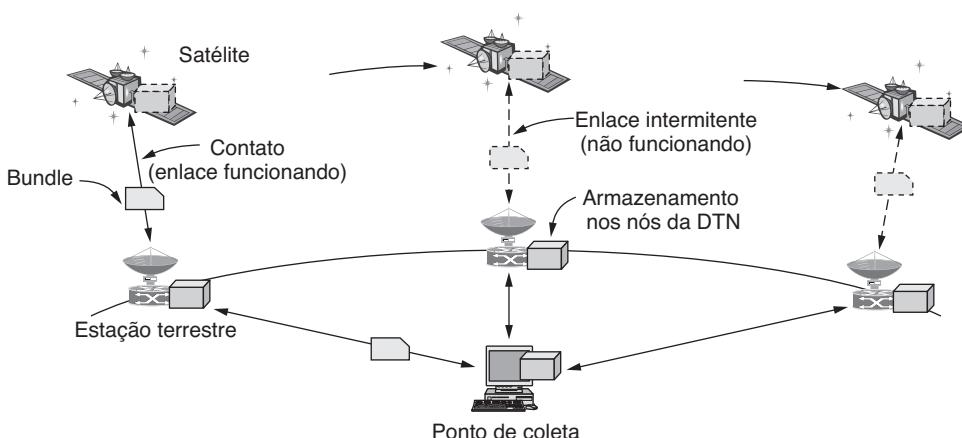



Figura 6.51 | Uso de uma DTN no espaço.

6.7.2 O protocolo Bundle

Para examinar mais de perto a operação das DTNs, agora veremos os protocolos IETF. As DTNs são um tipo de rede emergente e as DTNs experimentais têm usado diferentes protocolos, pois não existe a exigência de que os protocolos IETF sejam usados. Porém, eles são pelo menos um bom local para iniciar e destacam muitas das principais questões.

A pilha de protocolos DTN aparece na Figura 6.52. O principal protocolo é o **protocolo Bundle**, que é especificado na RFC 5050. Ele é responsável por aceitar mensagens da aplicação e enviá-las como um ou mais bundles por meio de operações store-carry-forward ao nó DTN de destino. Também fica evidente pela Figura 6.52 que o protocolo Bundle roda acima do nível do TCP/IP. Em outras palavras, TCP/IP pode ser usado sobre cada contato para mover bundles entre nós de DTN. Esse posicionamento aumenta a questão se o protocolo Bundle é um protocolo da camada de transporte ou um protocolo da camada de aplicação. Assim como o RTP, tomamos a posição de que, apesar de trabalhar sobre um protocolo de transporte, o protocolo Bundle está oferecendo um serviço de transporte para muitas aplicações, e por isso abordamos as DTNs neste capítulo.

Na Figura 6.52, vemos que o protocolo Bundle pode ser executado sobre outros tipos de protocolos, como UDP, ou ainda outros tipos de redes interligadas. Por exemplo, em uma rede espacial de enlaces pode haver atrasos muito grandes. O tempo de ida e volta entre a Terra e Marte pode facilmente chegar a 20 minutos, dependendo da posição relativa dos planetas. Imagine como as confirmações e retransmissões do TCP funcionarão em um enlace desse tipo, especialmente para mensagens relativamente curtas. Ele não se sairá bem. Em vez disso, outro protocolo que usa códigos de correção de erros poderia ser usado. Ou então, em redes de sensores que possuem recursos muito restritos, um protocolo mais leve que o TCP pode ser usado.

Como o protocolo Bundle é fixo, embora deva ser executado sobre uma grande variedade de transportes, deve haver uma lacuna na funcionalidade entre os protocolos. Essa lacuna é o motivo para a inclusão de uma camada de convergência na Figura 6.52. A camada de convergência é apenas uma camada de união que combina as interfaces dos protocolos que ela une. Por definição, existe uma camada de convergência diferente para cada transporte diferente na camada inferior. As camadas de convergência normalmente são encontradas em padrões para juntar protocolos novos aos existentes.

O formato das mensagens do protocolo Bundle aparece na Figura 6.53. Os diferentes campos nessas mensagens nos dizem algumas das principais questões que são tratadas pelo protocolo Bundle.

Cada mensagem consiste em um bloco primário, que pode ser considerado um cabeçalho, um bloco de carga útil para os dados e, opcionalmente, outros blocos, por exemplo, para transportar parâmetros de segurança. O bloco primário começa com um campo *Versão* (atualmente, 6), seguido por um campo *Flags*. Entre outras funções, os flags codificam uma classe de serviço para permitir que uma origem marque seus bundles como prioridade mais alta ou mais baixa, e outras solicitações de tratamento, como se o destino deverá confirmar o bundle ou não.

Depois aparecem os endereços, que destacam três partes interessantes do projeto. Assim como um campo de identificador de *Destino* e *Origem*, existe um identificador *Custodiante*. O custodiante é a parte responsável por verificar se o bundle foi entregue. Na Internet, o nó de origem normalmente é o custodiante, pois é o nó que retransmite se os dados não forem entregues ao destino. Porém, em uma DTN, o nó de origem pode nem sempre estar conectado, e pode não ter como saber se os dados foram entregues. As DTNs lidam com esse problema usando a noção de **transferência de custódia**, em que outro nó, mais próximo do destino, pode assumir a responsabilidade por verificar se os dados foram entregues com segurança. Por exemplo, se um bundle for armazenado em um avião para encaminhamento em outro momento e local, o avião pode se tornar o custodiante do bundle.

O segundo aspecto interessante é que esses identificadores *não* são endereços IP. Como o protocolo Bundle deve funcionar por uma série de transporte e redes interligadas, ele define seus próprios identificadores. Esses identificado-

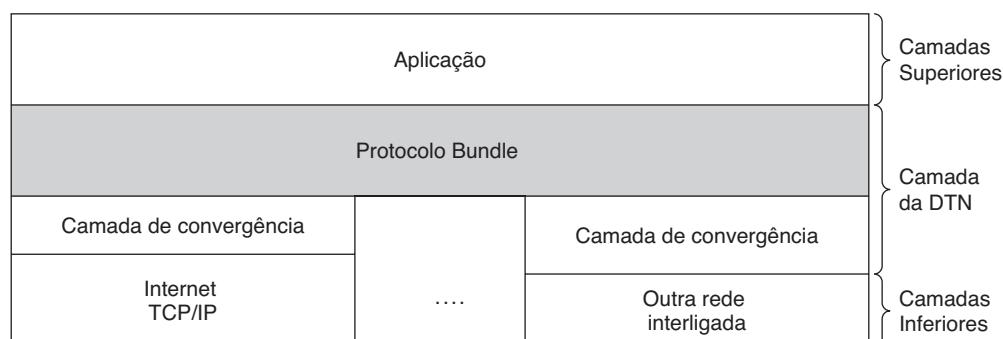



Figura 6.52 | Pilha de protocolos da rede tolerante a atrasos.

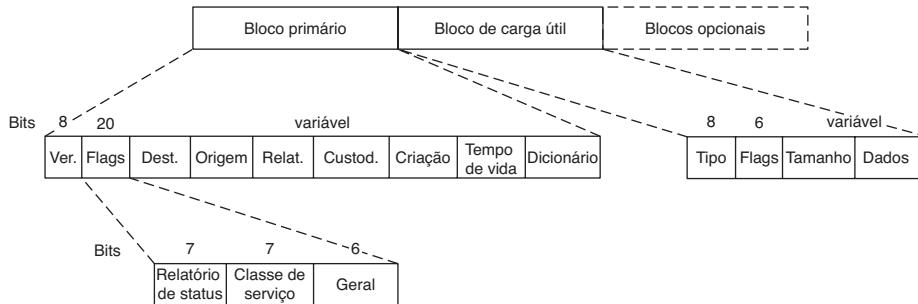



Figura 6.53 | Formato de mensagem do protocolo Bundle.

res na realidade são mais parecidos com nomes de alto nível, como URLs de página Web, do que endereços de baixo nível, como endereços IP. Eles dão às DTNs um aspecto de roteamento em nível de aplicação, como entrega de e-mail ou a distribuição de atualizações de software.

Um terceiro aspecto interessante é o modo como os identificadores são codificados. Há também um identificador de *Relatório* para mensagens de diagnóstico. Todos os identificadores são codificados como referências em um campo *Dicionário*, com tamanho variável. Este oferece compactação quando os nós custodiante ou relatório são os mesmos que a origem ou o destino. De fato, grande parte do formato da mensagem foi preparada visando à facilidade de extensão e à eficiência, usando uma representação compacta de campos de tamanho variável. A representação compacta é importante para enlaces sem fios e nós com recursos restritos, como em uma rede de sensores.

Em seguida vem um campo de *Criação*, que transporta o momento em que o bundle foi criado, junto com um número de sequência a partir da origem para ordenação, mais um campo de *Tempo de vida*, que informa o momento em que os dados do bundle não são mais úteis. Esses campos existem porque os dados podem ser armazenados por um longo período nos nós da DTN e é preciso haver algum modo de remover dados antigos da rede. Diferentemente da Internet, eles exigem que os nós da DTN tenham clocks vagamente sincronizados.

O bloco primário termina com o campo *Dicionário*. Depois vem o bloco de carga útil. Esse bloco começa com um pequeno campo de *Tipo* que o identifica como uma carga útil, seguido por um pequeno conjunto de *Flags* que descrevem opções de processamento. Depois vem o campo de *Dados*, precedido por um campo de *Tamanho*. Finalmente, pode haver outros blocos opcionais, como um bloco que transporta parâmetros de segurança.

Muitos aspectos das DTNs estão sendo explorados nas comunidades de pesquisa. Boas estratégias para o roteamento dependem da natureza dos contatos, como já mencionamos. O armazenamento de dados dentro da rede levanta outras questões. Agora, o controle do congestionamento precisa considerar o armazenamento nos nós como outro tipo de recurso que pode se esgotar. A falta de comunicação fim a fim também realça os problemas de segurança. Antes que um nó da DTN tome a custódia de um bundle, ele pode querer saber se o transmissor está autorizado a usar a rede e se o bundle provavelmente é desejado pelo destino. As soluções para esses problemas dependerão do tipo de DTN, pois as redes espaciais são diferentes das redes de sensores.

6.8 Resumo

A camada de transporte é a chave para a compreensão dos protocolos em camadas. Ela oferece vários serviços, dos quais o mais importante é um fluxo de bytes confiável, orientado a conexões e fim a fim, do transmissor até o receptor. Ela é acessada por meio de primitivas de serviço que permitem o estabelecimento, o uso e o encerramento de conexões. Uma interface comum da camada de transporte é oferecida pelos soquetes de Berkeley.

Os protocolos de transporte devem ser capazes de realizar o gerenciamento de conexões em redes não confiáveis. O estabelecimento de conexões é complicado pela existência de duplicatas de pacotes atrasados que podem reaparecer em momentos inoportunos. Para lidar com eles, são necessários handshakes de três vias para estabelecer conexões. O encerramento de uma conexão é um pouco mais fácil, mas ainda assim está longe de ser uma questão trivial, devido ao problema dos dois exércitos.

Mesmo quando a camada de rede é totalmente confiável, a camada de transporte tem muito trabalho a fazer. Ela deve manipular todas as primitivas de serviço, gerenciar as conexões e os timers, alocar largura de banda com controle de congestionamento e executar uma janela deslizante de tamanho variável para o controle de fluxo.

O controle de congestionamento deve alocar toda a largura de banda disponível entre fluxos concorrentes de forma imparcial, e deve acompanhar as mudanças no uso da rede. A lei de controle AIMD converge para uma alocação imparcial e eficiente.

A Internet tem dois protocolos de transporte importantes: UDP e TCP. O UDP é o protocolo não orientado a conexões, usado principalmente como um invólucro para pacotes IP, com o recurso adicional de multiplexar e demul-

tiplexar vários processos utilizando um único endereço IP. O UDP pode ser usado, por exemplo, em interações cliente-servidor, empregando RPC. Ele também pode ser usado na construção de protocolos em tempo real, como o RTP.

O principal protocolo de transporte da Internet é o TCP, que fornece um fluxo de bytes bidirecional confiável, com controle de congestionamento, com um cabeçalho de 20 bytes em todos os segmentos. Um grande trabalho tem sido realizado na tentativa de otimizar o desempenho do TCP, com os algoritmos de Nagle, Clark, Jacobson, Karn e outros.

O desempenho da rede normalmente é dominado pelo overhead de protocolos e de processamento de segmentos, e essa situação piora em velocidades mais altas. Os protocolos devem ser projetados para minimizar o número de segmentos e funcionar em caminhos com produto largura de banda-atraso alto. Para redes de gigabits, são necessários protocolos simples e processamento rápido.

A rede tolerante a atrasos oferece um serviço de entrega por redes que possuem conectividade ocasional ou longos atrasos em enlaces. Os nós intermediários armazenam, transportam e encaminham bundles de informações, de modo que eles finalmente são entregues, mesmo que não haja um caminho funcional do transmissor ao receptor em algum momento.

Problemas

1. Em nosso exemplo de primitivas de transporte da Tabela 6.1, a primitiva LISTEN é uma chamada de bloqueio. Ela é estritamente necessária? Caso não seja, explique como uma primitiva sem bloqueio poderia ser usada. Que vantagem isso ofereceria em relação ao esquema descrito no texto?
2. Primitivas do serviço de transporte consideram assimetria entre as duas extremidades durante o estabelecimento da conexão, com uma ponta (servidor) executando LISTEN enquanto a outra (cliente) executa CONNECT. Porém, em aplicações peer-to-peer típicas aos sistemas de compartilhamento de arquivos, como BitTorrent, todas as extremidades são peers. Não existe funcionalidade explícita de servidor ou cliente. Como as primitivas do serviço de transporte poderão ser usadas para montar essas aplicações peer-to-peer?
3. No modelo básico da Figura 6.3, parte-se do pressuposto de que os pacotes podem se perder na camada de rede e, por isso, devem ser confirmados individualmente. Suponha que a camada de rede seja 100 por cento confiável e nunca perca pacotes. Que mudanças, se houver, são necessárias na Figura 6.3?
4. Em ambas as partes do Quadro 6.1, existe um comentário de que o valor de *SERVER_PORT* deve ser igual no cliente e no servidor. Por que isso é tão importante?
5. No exemplo de servidor de arquivos da Internet (Quadro 6.1), a chamada de sistema à connect( ) no cliente pode falhar por algum motivo além de a fila de escuta estar cheia no servidor? Considere que a rede é perfeita.
6. Um critério para decidir se um servidor estará ativo o tempo inteiro ou se terá que iniciar por demanda usando um servidor de processos é com frequência o serviço usado. Você pode pensar em qualquer outro critério para tomar essa decisão?
7. Suponha que o esquema baseado em clock empregado na geração de números de sequência iniciais seja usado com um contador de clock de 15 bits de largura. O clock pulsa a cada 100 ms, e o tempo de vida máximo de cada pacote é de 60 s. Com que frequência a ressincronização deve ser feita:
   (a) Na pior das hipóteses?
   (b) Quando os dados consumirem 240 números de sequência/minuto?
8. Por que o tempo máximo de duração de um pacote *T* deve ser longo o suficiente para assegurar que não apenas o pacote mas também todas as suas confirmações tenham desaparecido?
9. Imagine que, em vez do handshake de três vias, tenha sido utilizado um handshake de duas vias para estabelecer uma conexão. Em outras palavras, a terceira mensagem não foi solicitada. É possível que agora ocorra um impasse? Forneça um exemplo ou mostre que não existe nenhum.
10. Imagine um problema genérico dos *n* exércitos, no qual um acordo entre dois quaisquer dos exércitos azuis é suficiente para a vitória. Existe algum protocolo que permita ao exército azul vencer?
11. Considere o problema da recuperação do funcionamento depois de panes nos hosts (Figura 6.15). Se o intervalo entre a gravação e o envio de uma confirmação, ou vice-versa, pode se tornar relativamente pequeno, quais são as duas melhores estratégias transmissor/receptor para minimizar a possibilidade de uma falha do protocolo?
12. Na Figura 6.17, suponha que um novo fluxo *E* seja acrescentado, seguindo de *R1* para *R2* para *R6*. Como a alocação de largura de banda max-min muda para os cinco fluxos?
13. Discuta as vantagens e desvantagens dos protocolos de janela deslizante.
14. Algumas outras políticas de imparcialidade no controle de congestionamento são Additive Increase Additive Decrease (AIAD), Multiplicative Increase Additive Decrease (MIAD) e Multiplicative Increase Multiplicative Decrease (MIMD). Discuta essas três políticas em termos de convergência e estabilidade.
15. Por que o UDP existe? Não teria sido suficiente deixar que os processos dos usuários enviassem pacotes IP brutos?
16. Considere um protocolo simples no nível da aplicação, elaborado sobre o UDP e que permite a um cliente recuperar um arquivo de um servidor remoto que reside em um endereço conhecido. Primeiro, o cliente envia uma solicitação com o nome do arquivo, e o servidor responde com uma se-

quência de pacotes de dados contendo partes diferentes do arquivo solicitado. Para assegurar confiabilidade e entrega em sequência, o cliente e o servidor utilizam um protocolo stop-and-wait. Ignorando a questão óbvia do desempenho, você percebe algum problema com esse protocolo? Pense cuidadosamente na possibilidade de panes em processos.

17. Um cliente transmite uma solicitação de 128 bytes a um servidor localizado a 100 km de distância sobre uma linha de fibra óptica de 1 gigabit. Qual é a eficiência da linha durante a chamada de procedimento remoto?
18. Considere a situação do problema anterior novamente. Calcule o menor tempo de resposta possível para a linha de 1 Gbps e para uma linha de 1 Mbps. Que conclusão é possível tirar desses dados?
19. Tanto o UDP quanto o TCP empregam números de portas para identificar a entidade de destino ao entregarem uma mensagem. Forneça duas razões pelas quais esses protocolos criaram uma nova ID abstrata (números de portas), em vez de usar IDs de processos, que já existiam quando esses protocolos foram projetados.
20. Várias implementações de RPC oferecem uma opção para o cliente usar RPC implementado sobre UDP ou RPC implementado sobre TCP. Sob que condições um cliente preferirá usar RPC sobre UDP e sob que condições ele preferirá usar RPC sobre TCP?
21. Considere duas redes, *N*1 e *N*2, que têm o mesmo atraso médio entre uma origem *A* e um destino *D*. Em *N*1, o atraso experimentado por diferentes pacotes é uniformemente distribuído com um atraso máximo de 10 segundos, enquanto em *N*2, 99 por cento dos pacotes experimentam menos de um segundo de atraso sem limite sobre o atraso máximo. Discuta como o RTP pode ser usado nesses dois casos para transmitir o fluxo de áudio/vídeo ao vivo.
22. Qual é o tamanho total da MTU mínima do TCP, incluindo o overhead do TCP e do IP, mas sem incluir o overhead da camada de enlace de dados?
23. A fragmentação e a remontagem de datagramas são tratadas pelo IP e são invisíveis para o TCP. Isso quer dizer que o TCP não tem de se preocupar com a chegada de dados na ordem errada?
24. O RTP é usado para transmitir áudio com qualidade de CD, que gera um par de amostras de 16 bits 44.100 vezes/s, uma amostra para cada um dos canais estereofônicos. Quantos pacotes por segundo o RTP deve transmitir?
25. Seria possível inserir o código do RTP no núcleo do sistema operacional, juntamente com o código do UDP? Explique sua resposta.
26. A um processo no host 1 foi atribuída a porta *p*, e a um processo no host 2 foi atribuída a porta *q*. É possível haver duas ou mais conexões TCP entre essas duas portas ao mesmo tempo?
27. Na Figura 6.31 vimos que, além do campo de *Confirmação* de 32 bits, existe um bit ACK na quarta palavra. Isso realmente acrescenta algo? Por quê?
28. A carga útil máxima de um segmento TCP é 65.495 bytes. Por que foi escolhido um número tão estranho?
29. Descreva duas maneiras de entrar no estado *SYN RCVD* da Figura 6.33.
30. Considere o efeito de usar a partida lenta em uma linha com um tempo de percurso de ida e volta de 10 ms e sem congestionamento. A janela de recepção tem 24 KB e o tamanho máximo do segmento é 2 KB. Quanto tempo é necessário para que a primeira janela completa possa ser enviada?
31. Suponha que a janela de congestionamento do TCP seja definida como 18 KB e que ocorra um timeout. Qual será o tamanho da janela se as próximas quatro rajadas de transmissão forem bem-sucedidas? Suponha que o tamanho máximo do segmento seja 1 KB.
32. Se o tempo de viagem de ida e volta no TCP, denominado RTT, for igual a 30 ms, e se as confirmações seguintes chegarem depois de 26, 32 e 24 ms, respectivamente, qual será a nova estimativa para *RTT* empregando-se o algoritmo de Jacobson? Considere $\alpha = 0{,}9$.
33. Uma máquina TCP está transmitindo janelas completas de 65.535 bytes através de um canal de 1 Gbps que tem um atraso de 10 ms em um dos sentidos. Qual é a vazão máxima que é possível alcançar? Qual é a eficiência da linha?
34. Qual é a maior velocidade da linha em que um host pode transmitir cargas úteis do TCP de 1.500 bytes com uma duração máxima de pacote de 120 segundos, sem que os números de sequência se repitam? Leve em conta o overhead do TCP, do IP e da Ethernet. Suponha que os quadros Ethernet possam ser enviados continuamente.
35. Para enfrentar as limitações do IP versão 4, um grande esforço teve de ser feito pela IETF, resultando no projeto do IP versão 6, e ainda existe relutância significativa na adoção dessa nova versão. Porém, nenhum esforço muito grande é necessário para resolver as limitações do TCP. Explique por que isso acontece.
36. Em uma rede com um tamanho máximo de segmento igual a 128 bytes, tempo máximo de duração de um segmento igual a 30 s e um número de sequência de 8 bits, qual é a taxa máxima de transferência de dados por conexão?
37. Suponha que você esteja medindo o tempo necessário para receber um segmento. Quando ocorrer uma interrupção, você lerá o clock do sistema em milissegundos. Quando o segmento tiver sido completamente processado, você lerá o clock mais uma vez. O valor 0 ms é medido 270 mil vezes, e 1 ms é medido 730 mil vezes. Quanto tempo é necessário para receber um segmento?
38. Uma CPU executa instruções a uma velocidade de 1.000 MIPS. Os dados podem ser copiados 64 bits de cada vez, sendo necessárias dez instruções para copiar cada palavra. Se um pacote recebido tiver de ser copiado duas vezes, esse sistema será capaz de tratar uma linha de 1 Gbps? Para simplificar, suponha que todas as instruções, mesmo aquelas que leem ou gravam na memória, funcionam a uma velocidade máxima de 1.000 MIPS.
39. Para contornar o problema da repetição dos números de sequência enquanto os pacotes antigos ainda existem, é possível utilizar números de sequência de 64 bits. Contudo, um cabo de fibra óptica pode utilizar uma velocidade

de 75 Tbps, pelo menos teoricamente. Qual é o tempo máximo de duração dos pacotes necessário para garantir que as redes de 75 Tbps do futuro não tenham problemas de repetição dos números de sequência, mesmo com números de sequência de 64 bits? Suponha que cada byte tenha seu próprio número de sequência, como acontece no TCP.

40. Na Seção 6.6.5, calculamos que uma linha de gigabits descarrega 80 mil pacotes/s no host, dando a ele apenas 6.250 instruções para processar e deixar metade do tempo da CPU para aplicações. Esse cálculo partiu do princípio de que cada pacote tem 1.500 bytes. Refaça os cálculos para um pacote com o tamanho dos pacotes ARPANET (128 bytes). Em ambos os casos, suponha que os tamanhos dos pacotes dados incluem todo o overhead.

41. Para uma rede de 1 Gbps operando por 4.000 km, o atraso é o fator limitante, e não a largura da banda. Considere uma MAN com uma distância média entre a origem e o destino de 20 km. Em qual taxa de dados o atraso do percurso de ida e volta devido à velocidade da luz é igual ao atraso da transmissão para um pacote de 1 KB?

42. Calcule o produto largura de banda-atraso para as redes a seguir: (1) T1 (1,5 Mbps), (2) Ethernet (10 Mbps), (3) T3 (45 Mbps) e (4) STS-3 (155 Mbps). Suponha um RTT de 100 ms. Lembre-se de que um cabeçalho do TCP tem 16 bits reservados para o *Tamanho de janela*. Quais são as implicações, de acordo com seus cálculos?

43. Qual é o produto largura de banda-atraso para um canal de 50 Mbps em um satélite geoestacionário? Se todos os pacotes forem de 1.500 bytes (incluindo o overhead), qual deverá ser o tamanho da janela, medido em pacotes?

44. O servidor de arquivos do Quadro 6.1 está longe de ser perfeito e poderiam ser feitos alguns aperfeiçoamentos. Faça as seguintes modificações:
    (a) Dê ao cliente um terceiro argumento que especifique um intervalo de bytes.
    (b) Adicione um flag do cliente -w que permita a gravação do arquivo no servidor.

45. Uma função comum que todos os protocolos de rede precisam é manipular mensagens. Lembre-se de que os protocolos manipulam mensagens acrescentando/removendo cabeçalhos. Alguns protocolos podem quebrar uma única mensagem em múltiplos fragmentos, e mais tarde juntar esses múltiplos fragmentos de volta em uma única mensagem. Para isso, projete e implemente uma biblioteca de gerenciamento de mensagem que ofereça suporte para criar uma nova mensagem, anexar um cabeçalho a uma mensagem, remover um cabeçalho de uma mensagem, quebrar uma mensagem em duas, combinar duas mensagens em uma e salvar uma cópia de uma mensagem. Sua implementação precisa minimizar a cópia de dados de um buffer para outro ao máximo possível. É fundamental que as operações que manipulam mensagens não toquem nos dados de uma mensagem, mas apenas manipulem ponteiros.

46. Projete e implemente um sistema de bate-papo que permita a comunicação entre vários grupos de usuários. Um coordenador de bate-papo reside em um endereço de rede conhecido, utiliza o UDP para comunicação com clientes de bate-papo, instala servidores de bate-papo para cada sessão de bate-papo e mantém um diretório de sessão de bate-papo. Existe um servidor de bate-papo por sessão. Um servidor de bate-papo usa o TCP para se comunicar com clientes. Um cliente de bate-papo permite que usuários iniciem, se reúnam e saiam de uma sessão de bate-papo. Projete e implemente o código do coordenador, do servidor e do cliente.

A camada de aplicação

Capítulo 7

Depois de passarmos por todas as camadas preliminares, chegamos àquela em que são encontradas todas as aplicações. As camadas inferiores à camada de aplicação têm a função de oferecer um serviço de transporte confiável, mas, na verdade, elas não executam nenhuma tarefa para os usuários. Neste capítulo, estudaremos algumas aplicações reais em redes.

No entanto, mesmo na camada de aplicação, há necessidade de protocolos de suporte, a fim de permitir que as aplicações funcionem. Da mesma forma, examinaremos um desses protocolos antes de iniciarmos o estudo das aplicações em si. O item em questão é o DNS, que cuida da nomenclatura na Internet. Depois disso, examinaremos três aplicações reais: correio eletrônico, a World Wide Web e, por fim, multimídia. Terminaremos o capítulo explicando melhor a distribuição de conteúdo, incluindo as redes peer-to-peer.

7.1 DNS — Domain Name System (Sistema de Nomes de Domínio)

Embora os programas possam se referir em teoria a páginas Web, caixas de correio e outros recursos que utilizam os endereços de rede (por exemplo, endereços IP) dos computadores em que estão armazenados, esses endereços são difíceis para as pessoas memorizarem. Além disso, navegar pelas páginas Web de uma empresa a partir de *128.111.24.41* significa que, se a empresa mudar o servidor Web para uma máquina diferente, com um endereço IP diferente, todos precisam ser informados sobre o novo endereço IP. Por isso, foram introduzidos nomes de alto nível, legíveis, a fim de desassociar os nomes das máquinas dos endereços dessas máquinas. Desse modo, o servidor Web da empresa poderia ser conhecido como www.cs.washington.edu independentemente do seu endereço IP. Todavia, a rede somente reconhece endereços numéricos; portanto, é necessário algum tipo de mecanismo para converter as strings ASCII em endereços de rede. Nas seções a seguir estudaremos como esse mapeamento é feito na Internet.

Na ARPANET havia simplesmente um arquivo, *hosts.txt*, que listava todos os nomes de computador e seus endereços IP. Toda noite, esse arquivo era acessado por todos os hosts no local em que era mantido. Para uma rede de algumas centenas de grandes máquinas com tempo compartilhado, essa estratégia funcionava bastante bem.

No entanto, bem antes que milhões de PCs estivessem conectados à Internet, todos perceberam que essa estratégia não poderia continuar a ser utilizada para sempre. Em algum momento, o arquivo acabaria por se tornar grande demais. No entanto, a razão mais importante é que poderia haver conflitos de nomes de hosts constantemente, a menos que os nomes fossem gerenciados de uma forma centralizada — algo totalmente fora de cogitação em uma enorme rede internacional, devido à carga e à latência. Para resolver esses problemas, foi criado o sistema de nomes de domínios, ou **DNS (Domain Name System)**, em 1983. Ele tem sido uma parte fundamental da Internet desde então.

A essência do DNS é a criação de um esquema hierárquico de atribuição de nomes baseado no domínio e de um sistema de banco de dados distribuído para implementar esse esquema de nomenclatura. Ele é mais usado para mapear nomes de hosts em endereços IP, mas também pode servir para outros objetivos. O DNS é definido nas RFCs 1034, 1035, 2181 e elaborado com mais detalhes em muitas outras.

Em resumo, o DNS é utilizado da forma descrita a seguir. Para mapear um nome em um endereço IP, um programa aplicativo chama um procedimento de biblioteca denominado **resolvedor** e repassa a ele o nome como um parâmetro. Vimos um exemplo de resolvedor, *gethostbyname*, na Figura 6.6. O resolvedor envia uma consulta contendo o nome para um servidor DNS local, que procura o nome e retorna uma resposta contendo o endereço IP ao resolvedor. Este, em seguida, retorna o endereço IP ao programa aplicativo que fez a chamada. As mensagens de consulta e resposta são enviadas como pacotes UDP. Munido do endereço IP, o programa pode então estabelecer uma conexão TCP com o host ou enviar pacotes UDP até ele.

7.1.1 O ambiente de nomes do DNS

Gerenciar um grande conjunto de nomes que está em mudança constante não é um problema de fácil resolução. Em um sistema postal, o gerenciamento de nomes é feito por meio do uso de letras que especificam (implícita ou explicitamente) o país, o estado ou a província, a cidade e a rua do destinatário. Graças ao uso desse tipo de endereçamento hierárquico, não há confusão entre o João da Silva que mora na Rua Barata Ribeiro, em São Paulo, e o João da Silva que mora na Rua Barata Ribeiro, no Rio de Janeiro. O DNS funciona da mesma forma.

Para a Internet, o topo da hierarquia de nomes é controlado por uma organização chamada **ICANN** (**Internet Corporation for Assigned Names and Numbers**). A ICANN foi criada para essa finalidade em 1998, como parte do amadurecimento da Internet para uma abrangência mundial, econômica. Conceitualmente, a Internet é dividida em mais de 250 **domínios de nível superior**, em que cada domínio cobre muitos hosts. Um domínio é dividido em subdomínios, e estes são partidos ainda mais, e assim por diante. Todos esses domínios podem ser representados por uma árvore, como mostra a Figura 7.1. As folhas representam domínios que não possuem subdomínios (mas contêm máquinas, é claro). Um domínio de folha pode conter um único host, ou então pode representar uma empresa e conter milhares de hosts.

Existem dois tipos de domínios de nível superior: genéricos e de países. Os genéricos, listados na Tabela 7.1, incluem domínios originais da década de 80 e domínios introduzidos por meio de solicitações à ICANN. Outros domínios genéricos de alto nível serão acrescentados no futuro.

Os domínios de países incluem uma entrada para cada país, conforme a definição da ISO 3166. Os nomes de domínio de país internacionalizado que usa alfabeto não latino foram introduzidos em 2010. Esses domínios permitem que as pessoas nomeiem hosts em árabe, cirílico, chinês ou outros idiomas.

É fácil obter um domínio de segundo nível, como nome-da-empresa.com. Os domínios de nível superior são controlados pelos **registradores** apontados pela ICANN. Para obter um nome, basta ir até um registrador correspondente (nesse caso, *com*) para verificar se o nome desejado está disponível e não é marca registrada de outra pessoa. Se não houver nenhum problema, o solicitante pagará uma pequena taxa anual e conseguirá o nome.

Porém, à medida que a Internet se torna mais comercial e mais internacional, ela também passa a ser mais disputada, especialmente em questões relacionadas a nomes. Essa controvérsia inclui a própria ICANN. Por exemplo, a criação do domínio *xxx* levou vários anos e exigiu a solução de vários casos em tribunal. Colocar conteúdo adulto voluntariamente em seu próprio domínio é uma coisa boa ou ruim? (Algumas pessoas não queriam conteúdo adulto disponível de forma alguma na Internet, enquanto outras queriam colocá-lo em um domínio, de modo que filtros pudessem facilmente localizá-lo e impedir seu uso por crianças.) Alguns dos domínios são auto-organizados, enquanto outros têm restrições sobre quem pode obter um nome, conforme indicado na Tabela 7.1. Mas que restrições são apropriadas? Pense no domínio *pro*, por exemplo. Ele é adotado por profissionais qualificados. Mas quem é um profissional? Médicos e advogados sem dúvida são profissionais. Mas e fotógrafos autônomos, professores de música, mágicos, bombeiros hidráulicos, barbeiros, exterminadores, tatuadores, mercenários e prostitutas? Essas profissões são válidas? De acordo com quem?

Também há dinheiro envolvido nos nomes. Tuvalu (o país) vendeu uma licença em seu domínio *tv* por US$ 50 milhões, tudo porque o código do país é bastante adequado para anunciar sites de televisão. Praticamente toda palavra comum (em inglês) é usada no domínio *com*, além dos erros de grafia mais usuais. Experimente pesquisar utensílios domésticos, animais, plantas, partes do corpo etc. A prática de registrar um domínio apenas para esperar e vendê-lo a uma parte interessada por um preço muito mais alto tem até mesmo um nome. Isso se chama **cybersquatting**. Muitas empresas que foram lentas quando a era da Internet começou descobriram que seus nomes de domínio óbvios já haviam sido usados quando tentaram adquiri-los. Em geral, desde que nenhuma marca registrada esteja sendo violada e nenhuma fraude seja envolvida, os nomes são atribuídos a quem chegar primeiro. Apesar disso, políticas para resolver disputas de nomenclatura ainda estão sendo preparadas.

Cada domínio tem seu nome definido pelo caminho ascendente entre ele e a raiz (sem nome). Esses componentes são separados por pontos. Dessa forma, o departamento de engenharia da Cisco poderia ser eng.cisco.com, em vez de um nome no estilo UNIX, como /com/sun/eng. Observe que essa nomenclatura hierárquica significa que eng.cisco.com.

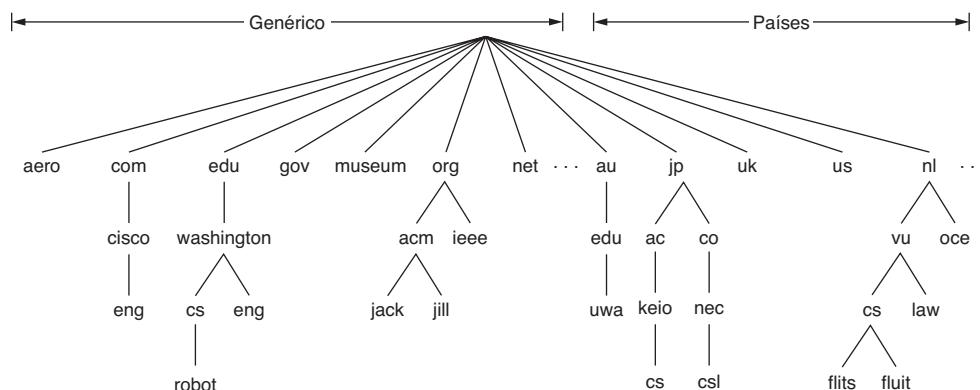



Figura 7.1 | Uma parte do espaço de nomes de domínios da Internet.

Domínio	Uso intencionado	Data de início	Restrito?
com	Comercial	1985	Não
edu	Instituições educacionais	1985	Sim
gov	Governo	1985	Sim
int	Organizações internacionais	1988	Sim
mil	Militares	1985	Sim
net	Provedores de rede	1985	Não
org	Organizações não lucrativas	1985	Não
aero	Transporte aéreo	2001	Sim
biz	Empresas	2001	Não
coop	Cooperativas	2001	Sim
info	Informativos	2002	Não
museum	Museus	2002	Sim
name	Pessoas	2002	Não
pro	Profissionais	2002	Sim
cat	Catalão	2005	Sim
jobs	Empregos	2005	Sim
mobi	Dispositivos móveis	2005	Sim
tel	Detalhes de contato	2005	Sim
travel	Indústria de viagens	2005	Sim
xxx	Indústria do sexo	2010	Não

Tabela 7.1 | Domínios genéricos de nível superior.

não entra em conflito com um possível uso de *eng* em eng.washington.edu., que poderia ser usado pelo departamento de língua inglesa da Universidade de Washington.

Os nomes de domínios podem ser absolutos ou relativos. Um nome de domínio absoluto sempre termina com um ponto (por exemplo, eng.sun.com.), ao contrário de um nome de domínio relativo. Os nomes relativos têm de ser interpretados em algum contexto para determinar exclusivamente seu verdadeiro significado. Em ambos os casos, um nome de domínio se refere a um nó específico da árvore e a todos os nós abaixo dele.

Os nomes de domínios não fazem distinção entre letras maiúsculas e minúsculas. Os nomes de componentes podem ter até 63 caracteres, e os nomes de caminhos completos não podem exceder 255 caracteres.

Em princípio, os domínios podem ser inseridos na árvore em domínios genéricos ou de país. Por exemplo, cs.washington.edu poderia ser igualmente listado sob o domínio de país *us* como cs.washington.wa.us. Contudo, na prática, quase todas as organizações dos Estados Unidos estão sob um domínio genérico e, praticamente, todas fora dos Estados Unidos estão sob o domínio de seu país. Não existe regra contra o registro sob dois domínios de nível superior. Grandes empresas normalmente fazem isso (por exemplo, sony.com, sony.net e sony.nl).

Cada domínio controla como serão alocados todos os domínios abaixo dele. Por exemplo, o Japão tem os domínios *ac.jp* e *co.jp*, que espelham *edu* e *com*. A Holanda não faz essa distinção e coloca todas as organizações diretamente sob *nl*. Assim, os três domínios a seguir representam departamentos de ciência da computação de universidades:

1. cs.washington.edu (Universidade de Washington, nos Estados Unidos);
2. cs.vu.nl (Vrije Universiteit, na Holanda);
3. cs.keio.ac.jp (Keio University, no Japão).

Para que um novo domínio seja criado, é necessária a permissão do domínio no qual ele será incluído. Por exemplo, se o grupo VLSI tiver começado na Universidade de Washington e quiser ser conhecido como vlsi.cs.washington.edu, ele precisará da permissão de quem gerencia cs.washington.edu. Da mesma forma, se uma nova universidade for licenciada, digamos a University of Nor-

thern South Dakota, ela terá de solicitar ao gerente do domínio *edu* que lhe atribua o domínio unsd.edu (se estiver disponível). Dessa forma, os conflitos de nomes são evitados e cada domínio pode controlar seus subdomínios. Uma vez que um novo domínio tenha sido criado e registrado, ele poderá criar subdomínios, tais como cs.unsd.edu, sem que seja necessária a permissão de alguém que esteja em um nível mais alto na árvore.

A atribuição de nomes leva em consideração as fronteiras organizacionais, e não as redes físicas. Por exemplo, mesmo que os departamentos de ciência da computação e de engenharia elétrica estejam no mesmo prédio e compartilhem a mesma LAN, eles poderão ter domínios distintos. Da mesma forma, mesmo que o departamento de ciência da computação esteja dividido em dois prédios, normalmente todos os hosts instalados em ambos pertencerão ao mesmo domínio.

7.1.2 Registros de recursos (RRs)

Todo domínio, seja um único host, seja um domínio de nível superior, pode ter um conjunto de **registros de recursos** associado a ele. Esses registros são o banco de dados DNS. Para um único host, o registro de recurso mais comum é apenas seu endereço IP, mas também existem muitos outros tipos. Quando um resolvedor repassa um nome de domínio ao DNS, o que ele obtém são os registros de recursos associados àquele nome. Portanto, a principal função do DNS é mapear nomes de domínios em registros de recursos.

Um registro de recurso é uma tupla de cinco campos. Apesar de serem codificados em binário para proporcionar maior eficiência, na maioria das exposições os registros de recursos são mostrados como texto ASCII, uma linha para cada registro de recurso. O formato que utilizaremos é:

Nome_domínio Tempo_de_vida Classe Tipo Valor

Nome_domínio informa o domínio ao qual esse registro se aplica. Normalmente, existem muitos registros para cada domínio, e cada cópia do banco de dados armazena informações sobre vários domínios. Assim, esse campo é a chave de pesquisa primária utilizada para atender às consultas. A ordem dos registros no banco de dados não é significativa.

Tempo_de_vida fornece uma indicação da estabilidade do registro. As informações muito estáveis são definidas com um número alto, como 86.400 (o número de segundos em 1 dia). As informações muito voláteis recebem um número baixo, como 60 (1 minuto). Voltaremos a esse ponto mais adiante, quando discutirmos o caching.

O terceiro campo de cada registro de recurso é *Classe*. No caso de informações relacionadas à Internet, ele é sempre *IN*. Para informações não relacionadas à Internet, podem ser empregados outros códigos; porém, estes raramente são encontrados na prática.

O campo *Tipo* informa qual é o tipo do registro. Os tipos mais importantes estão listados na Tabela 7.2.

Um registro *SOA* fornece o nome da principal fonte de informações sobre a zona do servidor de nomes (descrita a seguir), o endereço de correio eletrônico do administrador, um número de série exclusivo e diversos flags e timeouts.

O tipo de registro mais importante é o *A* (Address). Ele contém um endereço IPv4 de 32 bits de algum host. O registro *AAAA* correspondente, ou 'A quádruplo', mantém um endereço IPv6 de 128 bits. Cada host da Internet deve ter pelo menos um endereço IP, de forma que outras máquinas possam se comunicar com ele. Alguns hosts têm duas ou mais interfaces de rede; nesse caso, eles terão dois ou mais registros de recurso do tipo *A* ou *AAAA*. Consequentemente, o DNS pode retornar vários endereços para um único nome.

Tipo	Significado	Valor
SOA	Início de autoridade	Parâmetros para essa zona
A	Endereço IPv4 de um host	Inteiro de 32 bits
AAAA	Endereço IPv6 de um host	Inteiro de 128 bits
MX	Troca de mensagens de correio	Prioridade, domínio disposto a aceitar correio eletrônico
NS	Servidor de nomes	Nome de um servidor para este domínio
CNAME	Nome canônico	Nome de domínio
PTR	Ponteiro	Nome alternativo de um endereço IP
SPF	Estrutura de política do transmissor	Codificação de texto da política de envio de mensagens de correio
SRV	Serviço	Host que o oferece
TXT	Texto	Texto ASCII descritivo

Tabela 7.2 | Os principais tipos de registros de recursos.

Um tipo de registro comum é o *MX*. Ele especifica o nome do host preparado para aceitar mensagens de correio eletrônico para o domínio especificado. O registro *MX* é utilizado porque nem toda máquina está preparada para aceitar correio eletrônico. Se alguém quiser enviar correio eletrônico para bill@microsoft.com, o host transmissor precisará encontrar um servidor de correio em microsoft.com que esteja disposto a aceitar correio eletrônico. O registro *MX* pode fornecer essa informação.

Outro tipo de registro importante é o *NS*. Ele especifica um servidor de nomes para o domínio ou subdomínio. Ele é um host que tem uma cópia do banco de dados para um domínio e é usado como parte do processo de pesquisa de nomes, o que veremos adiante.

Os registros *CNAME* permitem a criação de nomes alternativos. Por exemplo, uma pessoa familiarizada com a Internet em geral que deseja enviar uma mensagem para alguém cujo nome de login seja *paul* no departamento de ciência da computação do MIT poderá imaginar que paul@cs.mit.edu seja o endereço correto. Na realidade, esse endereço não servirá, pois o domínio do departamento de ciência da computação do MIT é csail.mit.edu. No entanto, o MIT poderia criar uma entrada *CNAME* para orientar pessoas e programas na direção correta. Uma entrada como essa poderia executar essa função:

```
cs.mit.edu    86400    IN    CNAME    csail.mit.edu
```

A exemplo de *CNAME*, *PTR* indica outro nome. No entanto, ao contrário de *CNAME*, que, na verdade, é apenas um apelido de um nome atribuído originalmente [ou seja, CNAME permite substituir uma string (apelido) por outra (um nome canônico)], *PTR* é um tipo de dado comum do DNS, cuja interpretação depende do contexto no qual se encontra. Na prática, essa entrada é quase sempre usada para associar um nome a um endereço IP, a fim de permitir pesquisas de endereços IP e retornar o nome da máquina correspondente. Essas pesquisas são chamadas **pesquisas inversas**.

SRV é um tipo de registro mais novo, que permite que um host seja identificado para determinado serviço em um domínio. Por exemplo, o servidor Web para cs.washington.edu poderia ser identificado como cockatoo.cs.washington.edu. Esse registro generaliza o registro *MX* que realiza a mesma tarefa, mas é apenas para servidores de correio eletrônico.

SPF também é um tipo de registro mais novo. Ele permite que um domínio codifique informações sobre quais máquinas no domínio enviarão correio eletrônico ao restante da Internet. Isso ajuda as máquinas receptoras a verificar se o correio é válido. Se ele estiver sendo recebido de uma máquina que se chama *dodgy*, mas os registros de domínio disserem que o correio só será enviado do domínio por uma máquina chamada *smtp*, é provável que esse correio seja forjado.

Por fim, os registros *TXT* foram fornecidos originalmente para permitir que os domínios se identificassem de forma arbitrária. Hoje em dia, eles normalmente codificam informações legíveis à máquina, em geral a informação *SPF*.

Por fim, chegamos ao campo *Valor*. Esse campo pode ser um número, um nome de domínio ou uma string ASCII. A semântica dependerá do tipo de registro. Na Tabela 7.2, é mostrada uma breve descrição dos campos *Valor* de cada um dos principais tipos de registros.

Como exemplo do tipo de informação que se pode encontrar no banco de dados DNS de um domínio, observe o Quadro 7.1. Esse quadro ilustra parte de um banco de dados (hipotético) para o domínio cs.vu.nl mostrado na Figura 7.1. O banco de dados contém sete tipos de registros de recursos.

A primeira linha destinada a comentários do Quadro 7.1 apresenta algumas informações básicas sobre o domínio, que não nos interessarão em detalhes. As duas linhas seguintes mostram a primeira e a segunda opções para a entrega de mensagens de correio eletrônico enviadas para pessoa@cs.vu.nl. A entrada *zephyr* (uma máquina específica) deve ser a primeira opção a ser experimentada. Se ela não servir, *top* será a próxima opção. A próxima linha identifica o servidor de nomes para o domínio como *star*.

Depois da linha em branco (que foi incluída para facilitar a leitura) há outras informando os endereços IP para *star*, *zephyr* e *top*. Em seguida, há um nome alternativo, www.cs.vu.nl, ou seja, um endereço que pode ser usado sem a necessidade de especificar uma máquina. A criação desse nome alternativo permite que cs.vu.nl modifique seu servidor da World Wide Web sem invalidar o endereço que as pessoas utilizam para acessá-lo. Há um argumento semelhante para ftp.cs.vu.nl.

A seção para a máquina *flits* lista dois endereços IP e três escolhas são dadas para o tratamento de correio eletrônico enviado a flits.cs.vu.nl. A primeira escolha é naturalmente o próprio *flits*, mas se ele estiver fora do ar, *zephyr* e *top* são a segunda e a terceira opções, respectivamente.

As três linhas seguintes contêm uma entrada típica para um computador — nesse caso, rowboat.cs.vu.nl. As informações fornecidas contêm o endereço IP e as caixas de correio principal e secundária. Em seguida, vem uma entrada para um computador que não é capaz de receber correio por si só, seguida de uma entrada para uma impressora conectada à Internet.

7.1.3 Servidores de nomes

Pelo menos em teoria, um único servidor de nomes poderia conter o banco de dados DNS inteiro e responder a todas as consultas referentes ao banco. Na prática, esse servidor ficaria tão sobrecarregado que seria inútil. Além disso, caso ele ficasse fora do ar, toda a Internet seria atingida.

Para evitar os problemas associados à presença de uma única fonte de informações, o espaço de nomes do DNS é dividido em **zonas** não superpostas. Uma forma possível

; Dados oficiais para cs.vu.nl				
cs.vu.nl.	86400	IN	SOA	star boss (9527,7200,7200,241920,86400)
cs.vu.nl.	86400	IN	MX	1 zephyr
cs.vu.nl.	86400	IN	MX	2 top
cs.vu.nl.	86400	IN	NS	star
star	86400	IN	A	130.37.56.205
zephyr	86400	IN	A	130.37.20.10
top	86400	IN	A	130.37.20.11
www	86400	IN	CNAME	star.cs.vu.nl
ftp	86400	IN	CNAME	zephyr.cs.vu.nl
flits	86400	IN	A	130.37.16.112
flits	86400	IN	A	192.31.231.165
flits	86400	IN	MX	1 flits
flits	86400	IN	MX	2 zephyr
flits	86400	IN	MX	3 top
rowboat		IN	A	130.37.56.201
		IN	MX	1 rowboat
		IN	MX	2 zephyr
little-sister		IN	A	130.37.62.23
laserjet		IN	A	192.31.231.216

Quadro 7.1 | Uma parte de um possível banco de dados DNS para *cs.vu.nl*.

de dividir o espaço de nomes da Figura 7.1 é mostrada na Figura 7.2. Cada zona contém uma parte da árvore.

A localização das fronteiras de uma zona fica a cargo de seu administrador. Essa decisão é tomada principalmente com base no número de servidores de nomes desejados, e onde. Por exemplo, na Figura 7.2, a Universidade de Washington tem uma zona para washington.edu que trata de eng.washington.edu, mas não de cs.washington.edu, que é uma zona separada com seus próprios servidores de nomes. Tal decisão pode ser tomada quando um departamento como língua inglesa não deseja ter seu próprio servidor de nomes, mas um departamento como ciência da computação sim.

Cada zona também está associada a um ou mais servidores de nomes. Estes são hosts que mantêm o banco de dados para a zona. Normalmente, uma zona terá um servidor de nomes primário, que recebe sua informação de um arquivo em seu disco, e um ou mais servidores de nomes secundários, que recebem suas informações do servidor de nomes primário. Para melhorar a confiabilidade, alguns dos servidores de nomes podem estar localizados fora da zona.

O processo de pesquisa de um nome e localização de um endereço é chamado **resolução de nomes**. Quando um resolvedor tem uma consulta sobre um nome de domínio, ele passa a consulta para um servidor de nomes local. Se o domínio buscado cair sob a jurisdição do servidor de nomes, como top.cs.vu.nl caindo sob cs.vu.nl, ele retornará os registros de recursos oficiais. Um **registro oficial** é aquele que vem da autoridade que controla o registro e, portanto, sempre está correto. Os registros oficiais contrastam com os **registros em cache**, que podem estar desatualizados.

O que acontece quando o domínio é remoto, como quando flits.cs.vu.nl deseja encontrar o endereço IP de robot.cs.washington.edu em UW (Universidade de Washington)? Nesse caso, e se não houver informações sobre o domínio disponíveis localmente em cache, o servidor de nomes inicia uma consulta remota. Essa consulta segue o processo mostrado na Figura 7.3. A etapa 1 mostra a consulta que é enviada ao servidor de nomes local. Ela contém o nome de domínio buscado, o tipo (*A*) e a classe (*IN*).

A próxima etapa é começar no topo da hierarquia de nomes pedindo a um dos **servidores de nomes raiz**. Esses servidores de nomes têm informações sobre cada domínio de alto nível. Isso pode ser visto na etapa 2 da Figura 7.3. Para entrar em contato com um servidor-raiz, cada servidor

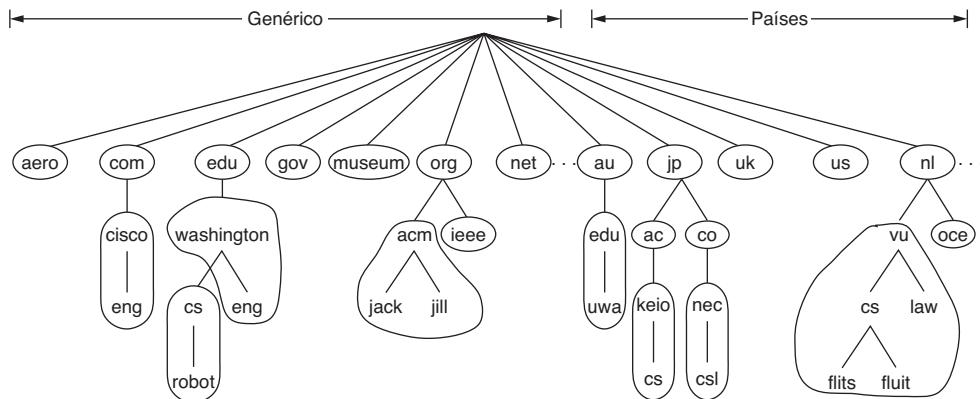



Figura 7.2 | Parte do espaço de nomes do DNS dividido em zonas (que estão circuladas).

de nomes precisa ter informações sobre um ou mais servidores de nomes raiz. Essa informação normalmente está presente em um arquivo de configuração do sistema que é carregado no cache DNS quando o servidor DNS é iniciado. Essa é simplesmente uma lista de registros *NS* para a raiz e os registros *A* correspondentes.

Existem 13 servidores de nomes raiz, de um modo pouco criativo chamados a-root-servers.net a m.root-servers.net. Cada servidor-raiz poderia ser logicamente um único computador. Porém, como a Internet inteira depende dos servidores-raiz, eles são computadores poderosos e altamente replicados. A maioria dos servidores está presente em vários locais geográficos e alcançados por meio de roteamento anycast, em que um pacote é entregue para a próxima instância de um endereço de destino; descrevemos o anycast no Capítulo 5. A replicação melhora a confiabilidade e o desempenho.

O servidor de nomes raiz provavelmente não saberá o endereço de uma máquina em UW, e provavelmente também não conhece o servidor de nomes para UW. Mas ele precisa conhecer o servidor de nomes para o domínio *edu*, em que cs.washington.edu está localizado. Ele retorna o nome e endereço IP para a parte da resposta na etapa 3.

O servidor de nomes local, então, continua sua busca. Ele envia a consulta inteira para o servidor de nomes edu (a.edu-servers.net). Esse servidor de nomes retorna um servidor de nomes para UW. Isso é visto nas etapas 4 e 5. Mais próximo agora, o servidor de nomes local envia a consulta para o servidor de nomes UW (etapa 6). Se o nome de domínio que está sendo buscado estivesse no departamento de língua inglesa, a resposta seria encontrada, pois a zona UW inclui o departamento de língua inglesa. Mas o departamento de ciência da computação decidiu manter seu próprio servidor de nomes. A consulta retorna o nome e endereço IP do servidor de nomes de ciência da computação de UW (etapa 7).

Finalmente, o servidor de nomes local consulta o servidor de nomes de ciência da computação de UW (etapa 8). Esse servidor é oficial para o domínio cs.washington.edu, de modo que deve ter a resposta. Ele retorna a resposta final (etapa 9), que o servidor de nomes local encaminha como resposta para flits.cs.vu.nl (etapa 10). O nome foi resolvido.

Você pode explorar esse processo usando ferramentas-padrão como o programa *dig* que está instalado na maioria dos sistemas UNIX. Por exemplo, digitar

```
dig@a.edu-servers.net robot.cs.washington.edu
```

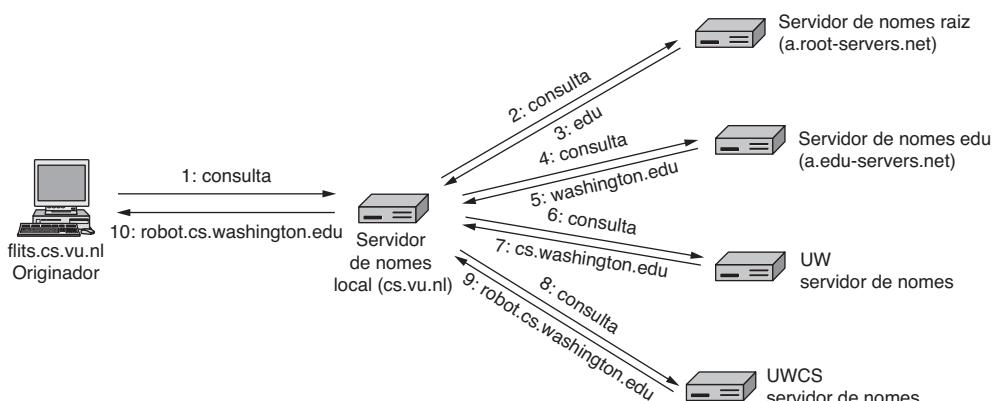



Figura 7.3 | O modo como um resolvedor procura um nome remoto em dez etapas.

motivará o envio de uma consulta para robot.cs.washington.edu ao servidor de nomes a.edu-servers.net e a impressão do resultado. Isso lhe mostrará a informação obtida na etapa 4 no exemplo anterior, e você aprenderá o nome e o endereço IP dos servidores de nomes UW.

Há três pontos técnicos a discutir sobre esse longo cenário. Primeiro, dois mecanismos de consulta estão em ação na Figura 7.3. Quando o host *flits.cs.vu.nl* envia sua consulta ao servidor de nomes local, o servidor de nomes trata a resolução em favor de *flits* até que tenha a resposta desejada para retornar. Ele *não* retorna respostas parciais. Elas poderiam ser úteis, mas não representam o que a consulta estava procurando. Esse mecanismo é chamado **consulta recursiva**.

Por outro lado, o servidor de nomes raiz (e cada servidor de nomes subsequente) não continua recursivamente a consulta para o servidor de nomes local. Ele apenas retorna uma resposta parcial e prossegue para próxima consulta. O servidor de nomes local é responsável por continuar a resolução emitindo outras consultas. Esse mecanismo é chamado **consulta iterativa**.

Uma resolução de nome pode envolver os dois mecanismos, como esse exemplo mostrou. Uma consulta recursiva sempre pode parecer preferível, mas muitos servidores de nomes (especialmente o raiz) não cuidarão disso. Eles são muito ocupados. As consultas iterativas colocam o peso sobre o originador. O raciocínio para o servidor de nomes local que dá suporte a uma consulta recursiva é que ele está fornecendo um serviço para os hosts em seu domínio. Esses hosts não precisam ser configurados para usar um servidor de nomes completo, apenas alcançar o servidor local.

O segundo ponto é o caching. Todas as respostas, incluindo todas as parciais retornadas, são mantidas em cache. Desse modo, se outro host cs.vu.nl procurar robot.cs.washington.edu, a resposta já será conhecida. Melhor ainda, se um host consultar um host diferente no mesmo domínio, digamos, galah.cs.washington.edu, a consulta poderá ser enviada diretamente para o servidor de nomes oficial. De modo semelhante, as consultas por outros domínios em washington.edu podem começar diretamente do servidor de nomes washington.edu. O uso de respostas em cache reduz bastante as etapas de uma consulta e melhora o desempenho. O cenário original que esboçamos é, de fato, o pior caso que ocorre quando nenhuma informação útil é mantida em cache.

Porém, as respostas em cache não são oficiais, pois as mudanças feitas em cs.washington.edu não serão propagadas para todos os caches no mundo que podem conhecê-la. Por esse motivo, as entradas em cache não devem ter vida longa. Esse é o motivo para o campo *Tempo_de_vida* ser incluído em cada registro de recurso. Ele diz aos servidores de nomes remotos por quanto tempo manter os registros em cache. Se determinada máquina tiver tido o mesmo endereço IP por anos, pode ser seguro manter essa informação em cache por um dia. Para informações mais voláteis, pode ser mais seguro eliminar os registros após alguns segundos ou minutos.

A terceira questão é o protocolo de transporte usado para consultas e respostas. Ele é o UDP. As mensagens DNS são enviadas em pacotes UDP com um formato simples para consultas, respostas e servidores de nomes que podem ser usados para continuar a resolução. Não entraremos nos detalhes desse formato. Se nenhuma resposta chegar em um curto período de tempo, o cliente DNS repetirá a consulta, escolhendo outro servidor para o domínio após um pequeno número de tentativas. Esse processo é criado para lidar com o caso do servidor que ficou inoperante, bem como com o do pacote de consulta ou resposta que se perdeu. Um identificador de 16 bits é incluído em cada consulta e copiado para a resposta, de modo que um servidor de nomes pode combinar as respostas com a consulta correspondente, mesmo que várias consultas estejam pendentes ao mesmo tempo.

Embora sua finalidade seja simples, deve ficar claro que o DNS é um sistema distribuído, grande e complexo, que compreende milhões de servidores de nomes que trabalham juntos. Ele forma um elo importante entre os nomes de domínio legíveis aos humanos e os endereços IP das máquinas. E inclui replicação e caching para ganhar desempenho e confiabilidade, sendo projetado para ser altamente robusto.

Não abordamos a segurança, mas, como você poderia imaginar, a capacidade de mudar o mapeamento entre nome e endereço pode ter consequências devastadoras se isso for feito de forma maliciosa. Por esse motivo, extensões de segurança, chamadas DNSSEC, foram desenvolvidas para DNS. Vamos descrevê-las no Capítulo 8.

Há também demanda da aplicação para usar nomes de maneiras mais flexíveis, por exemplo, dando nome ao conteúdo e resolvendo para o endereço IP do host próximo que possui o conteúdo. Isso se encaixa no modelo de busca e downloading de um filme. É o filme que importa, e não o computador que tem uma cópia dele, de modo que tudo o que é necessário é o endereço IP de qualquer computador próximo que tenha uma cópia do filme. As redes de distribuição de conteúdo são uma forma de realizar esse mapeamento. Vamos descrever como elas se baseiam no DNS mais adiante neste capítulo, na Seção 7.5.

7.2 Correio eletrônico

O correio eletrônico, ou **e-mail**, como é chamado por muitos, já existe há mais de duas décadas. Mais rápido e mais barato que o correio tradicional, o e-mail tem sido uma aplicação popular desde os primeiros dias da Internet. Antes de 1990, ele era empregado principalmente nos meios acadêmicos. Durante os anos 1990, ficou conhecido para o público em geral e seu uso cresceu exponencialmente, até alcançar um número de mensagens de correio eletrônico en-

viadas por dia imensamente maior que o número de cartas remetidas pelo **correio convencional** (isto é, cartas escritas em papel). Outras formas de comunicação pela rede, como mensagens instantâneas e chamadas de voz sobre IP, tiveram seu uso bastante expandido durante a década passada, mas o e-mail continua sendo a maior força da comunicação na Internet. Ele é muito usado na indústria para a comunicação dentro da empresa, por exemplo, para permitir que funcionários espalhados pelo mundo inteiro cooperem em projetos complexos. Infelizmente, assim como o correio tradicional, a maior parte do e-mail (cerca de 9 entre 10 mensagens) é lixo ou **spam** (McAfee, 2010).

Como a maioria das outras formas de comunicação, o correio eletrônico desenvolveu suas próprias convenções e estilos. Em particular, é muito informal e tem baixa limitação de uso. Pessoas que nunca sonhariam telefonar ou mesmo escrever uma carta para alguém muito importante não hesitam nem um segundo em enviar uma mensagem de correio eletrônico escrita às pressas e sem cuidado. Eliminando a maior parte das dicas associadas a cargo, idade e sexo, os debates por e-mail normalmente focalizam o conteúdo, e não o status. Com o e-mail, uma ideia brilhante de um aluno iniciante pode ter mais impacto do que uma ideia tola de um vice-presidente executivo.

O correio eletrônico está repleto de elementos de jargão, como AP (a propósito), RCTR (rolando no chão de tanto rir) e EMHO (em minha humilde opinião). Muitas pessoas também empregam pequenos símbolos ASCII chamados **smileys**, começando com o famoso ":-)". Gire o livro em 90 graus no sentido horário se esse símbolo não lhe for familiar. Esse símbolo e outros **emoticons** ajudam a transmitir o tom da mensagem. Eles se espalharam para outras formas de comunicação resumidas, como mensagens instantâneas.

Os protocolos de e-mail também evoluíram durante o período de seu uso. Os primeiros sistemas de correio eletrônico consistiam simplesmente em protocolos de transferência de arquivos, com a convenção de que a primeira linha de cada mensagem (isto é, o arquivo) deveria conter o endereço do destinatário. À medida que o tempo passou, o e-mail saiu da transferência de arquivos e muitos recursos foram acrescentados, como a capacidade de enviar uma mensagem para uma lista de destinatários. A capacidade de multimídia se tornou importante na década de 90 para enviar mensagens com imagens e outro tipo de material não textual. Os programas para ler e-mail se tornaram muito mais sofisticados, passando de texto para interfaces gráficas com o usuário e acrescentando a capacidade de os usuários acessarem seu correio a partir de notebooks, onde quer que estivessem. Finalmente, com a prevalência do spam, os leitores de e-mail e os protocolos de transferência de correio agora precisam prestar atenção na localização e remoção de e-mail indesejado.

Em nossa descrição de e-mail, vamos focalizar o modo como as mensagens de e-mail são movidas entre os usuários, em vez da aparência dos programas leitores de e-mail. Apesar disso, depois de descrever a arquitetura geral, vamos começar com a parte do sistema de e-mail voltada para o usuário, pois ela é familiar à maioria dos leitores.

7.2.1 Arquitetura e serviços

Nesta seção, apresentaremos uma visão geral do que os sistemas de e-mail podem fazer e como eles estão organizados. A arquitetura do sistema de e-mail aparece na Figura 7.4. Ela consiste em dois tipos de subsistemas: os **agentes do usuário**, que permitem que as pessoas leiam e enviem mensagens, e os **agentes de transferência de mensagens**, que deslocam as mensagens da origem até o destino. Também vamos nos referir aos agentes de transferência de mensagens informalmente como **servidores de correio.**

O agente do usuário é um programa que oferece uma interface gráfica, ou às vezes uma interface baseada em texto e comando, que permite aos usuários interagir com o sistema de e-mail. Ele inclui um meio de compor mensagens e respostas de mensagens, exibir mensagens que chegam e organizar mensagens por arquivamento, pesquisa e descarte. O ato de enviar novas mensagens ao sistema de e-mail para entrega é chamado de **submissão de e-mail**.

Parte do processamento do agente do usuário pode ser feita automaticamente, antecipando o que o usuário deseja. Por exemplo, o e-mail que chega pode ser filtrado para extrair ou diminuir a prioridade de mensagens que provavelmente são spam. Alguns agentes do usuário incluem recursos avançados, como a criação de respostas de e-mail automáticas

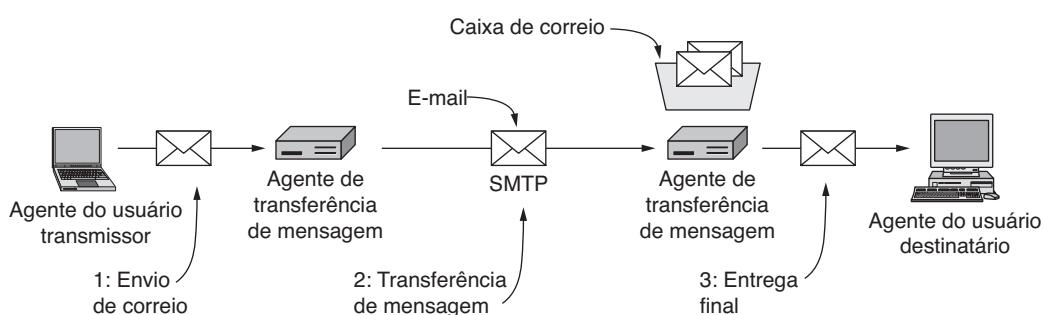



Figura 7.4 | Arquitetura do sistema de e-mail.

('Estou de férias e entrarei em contato assim que retornar'). Um agente do usuário é executado no mesmo computador em que um usuário lê seu e-mail. Esse é apenas outro programa e pode ser executado em apenas parte do tempo.

Os agentes de transferência de mensagem normalmente são processos do sistema. Eles trabalham em segundo plano nas máquinas servidoras de e-mail e sempre estão disponíveis. Seu trabalho é mover automaticamente o e-mail pelo sistema do remetente ao destinatário com o **SMTP** (**Simple Mail Transfer Protocol**). Essa é a etapa de transferência de mensagem.

O SMTP foi especificado originalmente como a RFC 821 e revisado para se tornar a atual RFC 5321. Ele envia e-mail pelas conexões e informa de volta o status de entrega e quaisquer erros. Existem várias aplicações em que a confirmação da entrega é importante e pode ainda ter significado legal ('Bem, meritíssimo, meu sistema de e-mail não é muito confiável, e creio que a intimação eletrônica se perdeu em algum lugar').

Os agentes de transferência de mensagem também implementam **listas de correspondência**, em que uma cópia idêntica de uma mensagem é entregue a todos em uma lista de endereços de e-mail. Outros recursos avançados são as cópias carbono (cc), as cópias carbono ocultas (cco), as mensagens de alta prioridade, as mensagens secretas (isto é, criptografadas), os destinatários alternativos caso o destinatário principal não esteja disponível no momento, e ainda a possibilidade de as secretárias lerem e responderem a correspondência de seus chefes.

Unindo os agentes do usuário e os agentes de transferência de mensagem estão os conceitos de caixas de correio e um formato-padrão para mensagens de e-mail. As **caixas de correio** armazenam o e-mail recebido para um usuário. Elas são mantidas pelos servidores de e-mail. Os agentes do usuário simplesmente apresentam aos usuários uma visão do conteúdo de suas caixas de correio. Para isso, os agentes enviam aos servidores de correio comandos para manipular as caixas de correio, inspecionar seu conteúdo, excluir mensagens e assim por diante. A recuperação do e-mail é a remessa final (etapa 3) na Figura 7.4. Com essa arquitetura, um usuário pode lançar mão de diferentes agentes do usuário em vários computadores para acessar uma única caixa de correio.

O e-mail é enviado entre os agentes de transferência de mensagem em um formato-padrão. O formato original, a RFC 822, foi revisado para a atual RFC 5322 e estendido com suporte para conteúdo de multimídia e texto internacional. Esse esquema é chamado MIME, e será discutido mais adiante. No entanto, as pessoas ainda se referem ao e-mail na Internet como RFC 822.

A ideia principal no formato da mensagem é a distinção entre o **envelope** e seu conteúdo. O envelope encapsula a mensagem. Ele contém toda a informação necessária para transportar a mensagem, como o endereço de destino, a prioridade e o nível de segurança, todos distintos da mensagem propriamente dita. Os agentes de transporte de mensagem utilizam o envelope para roteamento, assim como é feito pelo correio tradicional.

A mensagem dentro do envelope consiste em duas partes separadas: o **cabeçalho** e o **corpo**. O cabeçalho contém informações de controle para os agentes do usuário. O corpo é inteiramente endereçado para o destinatário humano. Nenhum dos agentes se importa muito com isso. Os envelopes e as mensagens são ilustradas na Figura 7.5.

Vamos estudar as partes dessa arquitetura com mais detalhes, examinando as etapas envolvidas no envio de e-mail de um usuário para outro. Essa jornada começa com o agente do usuário.

7.2.2 O agente do usuário

Um agente do usuário é um programa (às vezes chamado **leitor de e-mail**) que aceita uma série de comandos para compor, receber e responder as mensagens, bem como manipular caixas de correio. Existem muitos agentes do usuário populares, incluindo Gmail, do Google, Microsoft Outlook, Mozilla Thunderbird e Apple Mail. Eles podem variar bastante em sua aparência. A maioria dos agentes do usuário possui uma interface gráfica baseada em menu ou ícones, que exige um mouse ou uma interface de toque em dispositivos móveis menores. Agentes do usuário mais antigos, como Elm, mh e Pine, oferecem interfaces baseadas em texto e esperam comandos via caracteres do teclado. Funcionalmente, estes são os mesmos, pelo menos para mensagens de texto.

Os elementos típicos de uma interface de agente do usuário aparecem na Figura 7.6. Seu leitor de e-mail provavelmente será muito mais elegante, mas provavelmente tem funções equivalentes. Quando um agente do usuário for iniciado, ele normalmente apresentará um resumo das mensagens na caixa de correio do usuário. Geralmente, o resumo terá uma linha para cada mensagem em alguma ordem. Ele destaca os principais campos da mensagem, que são extraídos do envelope ou cabeçalho da mensagem.

Sete linhas de resumo aparecem no exemplo da Figura 7.6. As linhas usam os campos *From* (de), *Subject* (assunto) e *Received* (recebida em), nesta ordem, para mostrar quem enviou a mensagem, qual é o assunto e quando ela foi recebida. Toda a informação é formatada de uma maneira visualmente atraente, em vez de exibir o conteúdo literal dos campos da mensagem, mas ela é baseada nos campos da mensagem. Assim, as pessoas que não incluírem um campo *Subject* geralmente descobrirão que as respostas aos seus e-mails costumam não ter a prioridade mais alta.

Muitos outros campos ou indicações são possíveis. Os ícones ao lado dos assuntos da mensagem na Figura 7.6 poderiam indicar, por exemplo, e-mail não lido (o envelope), material anexado (o clipe de papel) e e-mail importante, pelo menos conforme julgado pelo emissor (o ponto de exclamação).

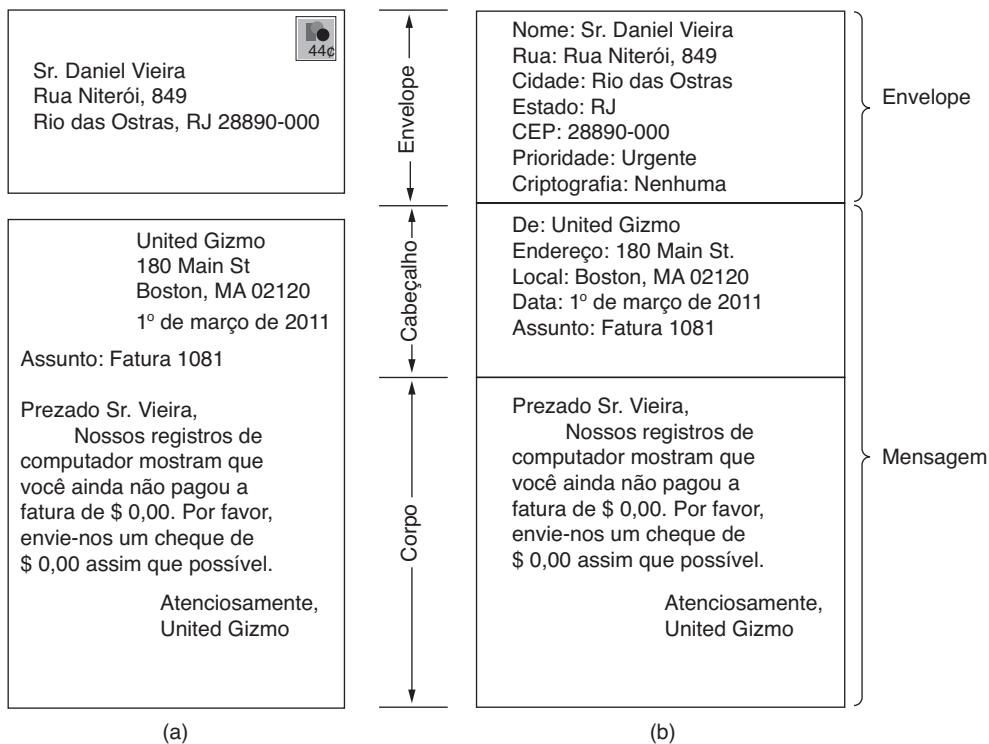



Figura 7.5 | Envelopes e mensagens. (a) Correio convencional. (b) Correio eletrônico.

Muitas ordens de classificação também são possíveis. A mais comum é ordenar as mensagens com base na hora em que foram recebidas, primeiro as mais recentes, com algo indicando se a mensagem é nova ou já foi lida pelo usuário. Os campos no resumo e a ordem de classificação podem ser personalizados pelo usuário, de acordo com suas preferências.

Os agentes do usuário também precisam ser capazes de exibir as mensagens que chegam conforme a necessidade, de modo que as pessoas possam ler seu e-mail. Quase sempre uma pequena prévia de uma mensagem é fornecida, como na Figura 7.6, para ajudar os usuários a decidir quando vão ler mais. As prévias podem usar pequenos ícones ou imagens para descrever o conteúdo da mensagem. Outro tipo de processamento da apresentação inclui reformatar as mensagens para que caibam na tela e traduzir ou converter o conteúdo para formatos mais convenientes (por exemplo, voz digitalizada para texto reconhecido).

Depois que a mensagem tiver sido lida, o usuário poderá decidir o que fazer com ela. Isso é chamado de **disposição de mensagem**. As opções incluem excluir a mensagem,

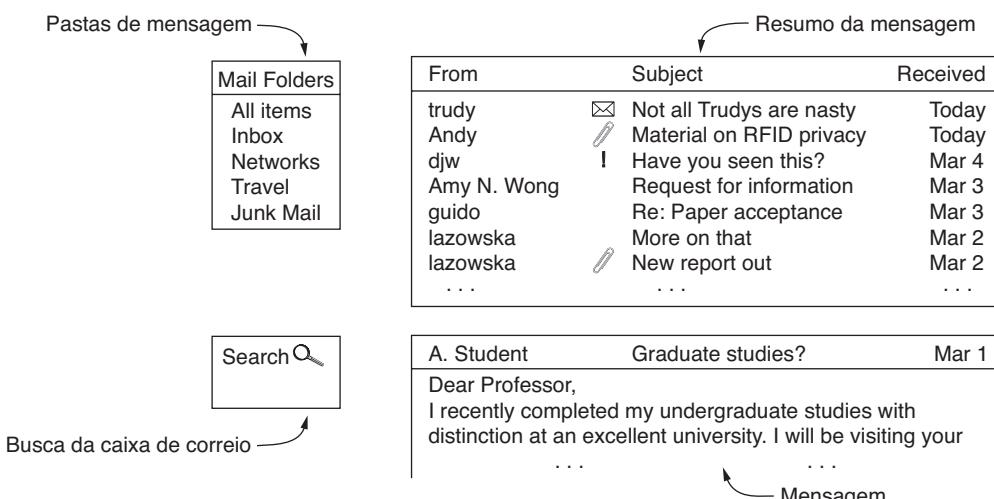



Figura 7.6 | Elementos típicos da interface do agente do usuário.

enviar uma resposta, encaminhar a outro usuário e manter a mensagem para referência futura. A maioria dos agentes do usuário pode gerenciar uma caixa de correio para o e-mail que chega com várias pastas para salvá-lo. As pastas permitem que o usuário salve mensagens de acordo com o remetente, assunto ou alguma outra categoria.

O arquivamento também pode ser feito automaticamente pelo agente do usuário, antes que este leia as mensagens. Um exemplo comum é que os campos e o conteúdo das mensagens sejam inspecionados e usados, junto com o feedback do usuário sobre mensagens anteriores, para determinar se uma mensagem provavelmente é um spam. Muitos ISPs e empresas executam software que rotula o e-mail como importante ou spam, para que o agente do usuário possa arquivá-lo na caixa de correio correspondente. O ISP e a empresa têm a vantagem de verificar o e-mail de muitos usuários e pode haver listas de divulgadores de spam conhecidos. Se centenas de usuários acabam de receber uma mensagem semelhante, ela provavelmente é um spam. Classificando previamente o e-mail que chega como 'provavelmente legítimo' e 'provavelmente spam', o agente do usuário pode evitar muito trabalho por parte dos usuários, separando o material bom do lixo.

E o spam mais popular? Ele é gerado por coleções de computadores infectados, chamados **botnets**, e seu conteúdo depende de onde você mora. Diplomas falsos são comuns na Ásia, e drogas baratas e outras ofertas de produtos duvidosos são comuns nos Estados Unidos. E-mails de contas bancárias não reivindicadas na Nigéria ainda estão circulando. Pílulas para aumentar várias partes do corpo são comuns em todos os lugares.

Outras regras de arquivamento podem ser elaboradas pelos usuários. Cada regra especifica uma condição e uma ação. Por exemplo, uma regra poderia dizer que qualquer mensagem recebida do chefe vai para uma pasta para leitura imediata e qualquer mensagem de uma lista de correspondência em particular vai para outra pasta, para ser lida mais tarde. Várias pastas aparecem na Figura 7.6. As pastas mais importantes são Inbox (ou caixa de entrada), para o e-mail que chega e não é arquivado em outro lugar, e Junk Mail (lixo), para mensagens que são consideradas spam.

Assim como as construções explícitas, como as pastas, os agentes do usuário agora oferecem ricas capacidades para pesquisar a caixa de correio. Esse recurso também aparece na Figura 7.6. Capacidades de busca permitem que usuários encontrem mensagens rapidamente, como a mensagem sobre 'onde comprar Vegemite no Brasil', que alguém enviou no mês passado.

O e-mail avançou muito desde os dias em que servia apenas para transferência de arquivos. Os provedores agora normalmente oferecem suporte para caixas de correio de até 1 GB de correio armazenado, detalhando as interações de um usuário por um longo período de tempo. O tratamento de correio sofisticado dos agentes do usuário, com buscas e formas automáticas de processamento, é o que torna possível administrar esses grandes volumes de e-mail. Para as pessoas que enviam e recebem milhares de mensagens por ano, essas ferramentas são muito valiosas.

Outro recurso útil é a capacidade de responder automaticamente a mensagens de alguma maneira. Uma resposta é encaminhar o e-mail que chega para um endereço diferente, por exemplo, um computador operado por um serviço de paging comercial, que transmite para o usuário por meio de rádio ou satélite e exibe a linha de assunto em seu pager. Essas **respostas automáticas** precisam ser executadas no servidor de e-mail, pois o agente do usuário pode não ser executado o tempo todo e só ocasionalmente poderá recuperar o e-mail. Devido a esses fatores, o agente do usuário não pode oferecer uma resposta verdadeiramente automática. Porém, a interface para respostas automáticas normalmente é apresentada pelo agente do usuário.

Um exemplo diferente de uma resposta automática é um **agente de férias**. Esse é um programa que examina cada mensagem que chega e envia ao remetente uma resposta insípida como. 'Oi. Estou de férias. Voltarei em 24 de agosto. Entrarei em contato quando retornar'. Essas respostas também podem especificar como tratar de questões urgentes nesse ínterim, outras pessoas a contatar para resolver problemas específicos etc. A maioria dos agentes de férias acompanha para quem estão sendo enviadas as respostas automáticas, para evitar enviar para a mesma pessoa uma segunda resposta. Porém, existem armadilhas com esses agentes. Por exemplo, não é aconselhável enviar uma resposta automática para uma grande lista de correspondência.

Agora, vejamos o cenário de um usuário enviando uma mensagem para outro. Um dos recursos básicos que os agentes do usuário oferecem, que ainda não discutimos, é a composição de mensagens. Isso envolve a criação de mensagens e respostas de mensagens, para enviá-las ao restante do sistema de e-mail para a remessa. Embora qualquer editor de textos possa ser usado para criar o corpo da mensagem, os editores normalmente são integrados ao agente do usuário, para que ele possa oferecer auxílio com o endereçamento e com os diversos campos de cabeçalho anexados a cada mensagem. Por exemplo, ao responder a uma mensagem, o sistema de e-mail pode extrair o endereço do remetente do e-mail que chegou e automaticamente inseri-lo no local apropriado na resposta. Outros recursos comuns são anexar um **bloco de assinatura** no final de uma mensagem, corrigir a grafia e calcular assinaturas digitais que mostrem que a mensagem é válida.

As mensagens enviadas para o sistema de e-mail têm um formato-padrão, que precisa ser criado a partir das informações fornecidas ao agente do usuário. A parte mais importante da mensagem para transferência é o envelope, e a parte mais importante do envelope é o endereço de destino. Esse endereço deve estar em um formato com o qual os agentes de transferência de mensagem possam lidar.

A forma esperada de um endereço é usuário@endereço-dns. Como estudamos o DNS anteriormente neste capítulo, não repetiremos esse material aqui. No entanto, vale a pena observar que existem outras formas de endereçamento. Em particular, os endereços **X.400** têm aspecto radicalmente diferente dos endereços DNS.

O X.400 é um padrão ISO para sistemas de tratamento de mensagem, que em certa época foi concorrente do SMTP. O SMTP venceu facilmente, embora os sistemas X.400 ainda sejam utilizados, principalmente fora dos Estados Unidos. Os endereços X.400 são compostos de pares *atributo = valor* separados por barras; por exemplo:

/C = US/ST = MASSACHUSETTS/L = CAMBRIDGE/
PA = 360 MEMORIAL DR./CN = KEN SMITH/

Esse endereço especifica um país, um estado, uma localidade, um endereço pessoal e um nome comum (Ken Smith). Muitos outros atributos são possíveis; portanto, você pode enviar uma mensagem de e-mail para alguém cujo endereço exato de e-mail você não conhece, desde que conheça outros atributos em número suficiente (por exemplo, empresa e cargo).

Embora os nomes X.400 sejam bem menos convenientes que os nomes DNS, a maioria dos sistemas de correio eletrônico tem nomes alternativos (às vezes chamados apelidos), que permitem aos usuários digitar ou selecionar o nome de uma pessoa e obter o endereço de e-mail apropriado. Consequentemente, em geral não é necessário digitar essas strings estranhas.

Um último ponto que abordaremos para o envio de e-mail são as listas de correspondência, que permitem que os usuários enviem a mesma mensagem para uma lista de pessoas com um único comando. Existem duas escolhas para o modo como a lista de correspondência é mantida. Ela pode ser mantida localmente, pelo agente do usuário. Nesse caso, o agente do usuário pode apenas enviar uma mensagem separada para cada destinatário desejado.

Como alternativa, a lista pode ser mantida remotamente em um agente de transferência de mensagem. As mensagens, então, serão expandidas no sistema de transferência de mensagem, com o efeito de permitir que vários usuários enviem para a lista. Por exemplo, se um grupo de observadores de pássaros tiver uma lista de correspondência chamada *birders* instalada no agente de transferência *meadowlark.arizona.edu*, qualquer mensagem enviada para *birders@meadowlark.arizona.edu* será roteada para a Universidade do Arizona e expandida para mensagens individuais para todos os membros dessa lista, onde quer que estejam no mundo inteiro. Os usuários dessa lista de correspondência podem não saber que essa é uma lista de correspondência. Ela poderia muito bem ser a caixa de correio pessoal do Prof. Gabriel Trinca Ferro.

7.2.3 Formatos de mensagens

Agora passaremos da interface do usuário para o formato das mensagens de correio eletrônico propriamente ditas. As mensagens enviadas pelo agente do usuário precisam ser colocadas em um formato-padrão, para ser tratadas pelos agentes de transferência de mensagem. Primeiro, estudaremos as mensagens básicas em código ASCII utilizando a RFC 5322, que é a última revisão do formato de mensagem original da Internet, descrito na RFC 822. Depois disso, veremos as extensões de multimídia aplicadas ao formato básico.

RFC 5322 — O formato de mensagem da Internet

As mensagens consistem em um envelope básico (descrito como parte do SMTP na RFC 5321), em alguns campos de cabeçalho, em uma linha em branco e no corpo da mensagem. Cada campo do cabeçalho consiste (logicamente) em uma única linha de texto ASCII contendo o nome do campo, um sinal de dois pontos e, quase sempre, um valor. A RFC 822 original foi projetada há décadas e não distinguia claramente os campos do envelope dos campos do cabeçalho. Embora ela tenha sido revista na RFC 5322, não foi possível refazê-la completamente, devido à sua utilização difundida. Em uso normal, o agente do usuário cria uma mensagem e a repassa ao agente de transferência de mensagens, que, em seguida, emprega alguns dos campos de cabeçalho para criar o envelope real, em uma mistura meio antiquada de mensagem e envelope.

Os principais campos do cabeçalho relacionados ao transporte de mensagens são listados na Tabela 7.3 O campo *To:* indica o endereço DNS do destinatário principal. Também é possível ter vários destinatários. O campo *Cc:* contém os endereços dos destinatários secundários (se houver). Em termos de entrega, não há distinção entre os destinatários principal e secundário. Trata-se de uma diferença inteiramente psicológica, importante apenas para as pessoas envolvidas, mas que não afeta o sistema de correio. O termo *Cc:* (cópia carbono) já está meio ultrapassado, pois os computadores não utilizam papel-carbono, mas é uma expressão consagrada pelo uso. O campo *Cco:* (cópia carbono oculta) é semelhante ao campo *Cc:*, exceto pelo fato de essa linha ser eliminada de todas as cópias enviadas aos destinatários principais e secundários. Esse recurso permite que as pessoas enviem cópias a terceiros sem que os destinatários principais e secundários saibam disso.

Os dois campos seguintes, *From:* e *Sender:*, informam quem escreveu e enviou a mensagem, respectivamente. Nem sempre esses campos conterão valores iguais. Por exemplo, um executivo pode escrever uma mensagem, mas na verdade sua secretária é quem a acaba transmitindo. Nesse caso, o executivo seria listado no campo *From:* e a secretária no campo *Sender:*. O campo *From:* é obrigatório, ao passo que *Sender:* pode ser omitido se for igual a *From:*. Esses campos são necessários caso a mensagem não possa ser entregue e tenha de ser devolvida ao remetente.

Cabeçalho	Significado
To:	O(s) endereço(s) de correio eletrônico do(s) destinatário(s) principal(is)
Cc:	O(s) endereço(s) de correio eletrônico do(s) destinatário(s) secundário(s)
Cco:	O(s) endereço(s) de correio eletrônico para cópias carbono ocultas
From:	A(s) pessoa(s) que criou(aram) a mensagem
Sender:	O endereço de e-mail do remetente
Received:	A linha incluída por cada agente de transferência ao longo da rota
Return-Path:	Pode ser usado para identificar um caminho de volta ao remetente

Tabela 7.3 | Campos do cabeçalho da RFC 5322 relacionados ao transporte da mensagem.

Uma linha contendo *Received:* é incluída por cada agente de transferência de mensagens ao longo do percurso. A linha contém a identidade do agente, a data e a hora em que a mensagem foi recebida e outras informações que podem ser usadas para localização de bugs no sistema de roteamento.

O campo *Return-Path:* é incluído pelo último agente de transferência de mensagens e seu objetivo é informar como voltar ao remetente. Em teoria, essas informações podem ser obtidas a partir de todos os cabeçalhos *Received:* (exceto pelo nome da caixa de correio do remetente), mas ele raramente é preenchido dessa forma e, em geral, contém apenas o endereço do remetente.

Além dos campos da Tabela 7.3, as mensagens RFC 5322 também podem conter uma variedade de campos de cabeçalho utilizados pelos agentes do usuário ou pelos destinatários. Os mais comuns estão listados na Tabela 7.4. A maior parte deles é autoexplicativa e não entraremos em detalhes sobre todos eles.

Às vezes o campo *Reply-To:* é usado quando nem a pessoa que redigiu nem a que enviou a mensagem querem ver a resposta. Por exemplo, um gerente de marketing escreve uma mensagem apresentando um novo produto aos clientes. A mensagem é enviada pela secretária, mas o campo *Reply-To:* lista o chefe do departamento de vendas, que pode responder às perguntas e receber pedidos do produto. Esse campo também é útil quando o remetente tem duas contas de correio eletrônico e deseja que a resposta vá para a outra conta.

O campo *Message-Id:* é um número gerado automaticamente, que é usado para vincular as mensagens (por exemplo, quando usado no campo *In-Reply-To:*) e para impedir a entrega duplicada.

O documento RFC 5322 menciona explicitamente que os usuários têm permissão de criar novos cabeçalhos para seu próprio uso. Por convenção, desde a RFC 822 esses cabeçalhos começam com a string *X*-. É certo que nenhum cabeçalho utilizará nomes começando com *X*- no futuro, a fim de evitar conflitos entre cabeçalhos oficiais e particulares. Às vezes, alguns estudantes pretensiosos criam campos como *X-Fruta-do-Dia:* ou *X-Doenca-da-Semana:*, que são válidos, mas nem sempre muito esclarecedores.

Depois dos cabeçalhos vem o corpo da mensagem. Os usuários colocam o que desejarem aqui. Algumas pessoas encerram suas mensagens com assinaturas elaboradas, incluindo citações de autores ou personalidades, declarações políticas e ressalvas de todos os tipos (por exemplo, a ABC Corporation não é responsável por minhas opiniões; na verdade, ela nem sequer as compreende).

Cabeçalho	Significado
Date:	A data e a hora em que a mensagem foi enviada
Reply-To:	O endereço de e-mail para onde as respostas devem ser enviadas
Message-Id:	O número exclusivo que será usado para fazer referência a essa mensagem posteriormente
In-Reply-To:	Message-Id da mensagem original correspondente a essa resposta
References:	Outras Message-Ids relevantes
Keywords:	Palavras-chave do usuário
Subject:	Pequeno resumo da mensagem apresentado em apenas uma linha

Tabela 7.4 | Alguns campos usados no cabeçalho de mensagem RFC 5322.

MIME – Multipurpose Internet Mail Extensions

Nos primórdios da ARPANET, o correio eletrônico consistia exclusivamente em mensagens de texto escritas em linguagem comum e expressas em código ASCII. Para esse ambiente, a RFC 822 fez tudo o que era preciso: especificou os cabeçalhos mas deixou o conteúdo inteiramente a cargo dos usuários. Nos anos 1990, o uso mundial da Internet e a demanda por conteúdo mais rico através do sistema de e-mail mostrou que isso não era mais adequado. Os problemas incluíam o envio e o recebimento de mensagens em idiomas com acentos (por exemplo, português e alemão); em alfabetos não latinos (por exemplo, hebraico e russo); em idiomas sem alfabetos (por exemplo, chinês e japonês); bem como mensagens que não contêm textos (por exemplo, áudio, imagens e documentos e programas binários).

A solução foi o desenvolvimento do **MIME (Multipurpose Internet Mail Extensions)**, que está sendo amplamente utilizado para mensagens de e-mail enviadas pela Internet, bem como para descrever o conteúdo para outras aplicações, como a navegação Web. O MIME é descrito nas RFCs 2045–2047, 4288, 4289 e 2049.

A ideia básica do MIME é continuar a usar o formato RFC 822 (a precursora da RFC 5322 na época em que o MIME foi proposto), mas incluir uma estrutura para o corpo da mensagem e definir regras para mensagens que não utilizam o código ASCII. Por manterem o formato 822, as mensagens no padrão MIME podem ser enviadas através da utilização dos agentes e protocolos de e-mail existentes no mercado (com base na RFC 821 na época, e na RFC 5321 agora). Só é necessário alterar os programas de envio e recebimento, o que os próprios usuários podem fazer.

O MIME define cinco novos cabeçalhos de mensagens, como mostra a Tabela 7.5. O primeiro deles informa ao agente do usuário receptor da mensagem não apenas que ele está lidando com uma mensagem MIME, como também a versão do padrão MIME que está sendo usada. Presume-se que qualquer mensagem que não contenha um cabeçalho *MIME-Version:* seja uma mensagem de texto simples escrita em linguagem comum (pelo menos uma usando apenas caracteres ASCII) e processada como tal.

O cabeçalho *Content-Description:* é uma string ASCII que informa o conteúdo da mensagem. Esse cabeçalho é necessário para que o destinatário saiba se vale a pena decodificar e ler a mensagem. Se a string informar 'Foto do ratinho da Bárbara' e a pessoa que receber a correspondência não for muito chegada a esses bichinhos, provavelmente a mensagem será descartada em vez de ser decodificada em uma fotografia colorida de alta resolução.

O cabeçalho *Content-Id:* identifica o conteúdo. Ele utiliza o mesmo formato do cabeçalho *Message-Id*.

Content-Transfer-Encoding: informa como o corpo da mensagem está codificado para transmissão através da rede. Um problema importante na época em que o MIME foi desenvolvido foi que os protocolos de transferência de e-mail (SMTP) esperavam mensagens ASCII em que nenhuma linha era superior a mil caracteres. Os caracteres ASCII utilizam 7 bits de cada byte com 8 bits. Os dados binários como programas executáveis e imagens utilizam todos os 8 bits de cada byte, assim como conjuntos de caracteres estendidos. Não havia garantia de que esses dados seriam transferidos com segurança. Logo, era preciso algum método de transportar dados binários que os tornasse parecidos com mensagens de e-mail ASCII comuns. As extensões ao SMTP desde o desenvolvimento do MIME permitem que dados binários de 8 bits sejam transferidos, embora até hoje os dados binários nem sempre possam passar pelo sistema de e-mail corretamente se não forem decodificados.

O MIME oferece cinco esquemas de codificação de transferência, mais um escape para novos esquemas — caso seja necessário. O esquema mais simples é de mensagens de texto ASCII. Os caracteres ASCII utilizam 7 bits e podem ser transportados diretamente pelo protocolo de e-mail, desde que nenhuma linha ultrapasse mil caracteres.

O esquema mais simples seguinte é igual ao anterior, mas utiliza caracteres de 8 bits, isto é, todos os valores de 0 a 255, inclusive. As mensagens que utilizam a codificação de 8 bits também devem aderir ao tamanho máximo de linha-padrão.

Depois existem as mensagens que utilizam a codificação binária. Essas mensagens são arquivos binários que não só utilizam todos os 8 bits, mas também não respeitam sequer o limite de linha de mil caracteres. Os programas executáveis estão nessa categoria. Hoje, os servidores de e-mail podem negociar para enviar dados em codificação binária (ou em 8 bits), passando para ASCII se os dois lados não aceitarem a extensão.

Cabeçalho	Significado
MIME-Version:	Identifica a versão do MIME
Content-Description:	String inteligível que identifica o conteúdo da mensagem
Content-Id:	Identificador exclusivo
Content-Transfer-Encoding:	Como o corpo da mensagem é codificado para transmissão
Content-Type:	Tipo e formato do conteúdo

Tabela 7.5 | Cabeçalhos de mensagem acrescentados pelo MIME.

A codificação ASCII de dados binários é chamada **codificação base64**. Nesse esquema, grupos de 24 bits são divididos em até quatro unidades de 6 bits, com cada unidade sendo enviada como um caractere ASCII válido. A codificação é 'A' para 0, 'B' para 1 e assim por diante, seguida pelas 26 letras minúsculas, pelos dez dígitos e concluindo + e / para 62 e 63, respectivamente. As sequências ==e = são usadas para indicar que o último grupo continha apenas 8 ou 16 bits, respectivamente. Os caracteres de retorno de cursor e avanço de linha são ignorados; portanto, podem ser inseridos à vontade para manter as linhas curtas. Os textos binários podem ser enviados com segurança usando esse esquema, embora de modo ineficiente. Essa codificação foi muito popular antes que os servidores de e-mail capazes de trabalhar com binário fossem implementados. Isso ainda é visto de modo geral.

Para as mensagens quase totalmente em ASCII, mas com alguns caracteres não pertencentes ao código ASCII, a codificação base64 se mostra um tanto ineficiente. Em vez disso, é utilizado um método conhecido como **codificação quoted-printable**. Trata-se apenas de um código ASCII de 7 bits, com todos os caracteres acima de 127 codificados como um sinal de igualdade seguido pelo valor do caractere como dois dígitos hexadecimais. Os caracteres de controle, algumas marcas de pontuação e símbolos matemáticos, bem como os espaços finais, também são codificados dessa forma.

Por fim, quando houver motivos válidos para não utilizar um desses esquemas, será possível especificar uma decodificação definida pelo usuário no cabeçalho *Content-Transfer-Encoding:*.

O último cabeçalho mostrado na Tabela 7.5 é na realidade o mais interessante. Ele especifica a natureza do corpo da mensagem e tem tido um impacto bem maior que o e-mail. Por exemplo, o conteúdo baixado da Web é rotulado com tipos MIME, de modo que o navegador sabe como apresentá-lo. O mesmo acontece com o streaming de mídia e transportes em tempo real, como VoIP.

Inicialmente, são definidos sete tipos MIME na RFC 1521, e cada um deles tem um ou mais subtipos. O tipo e o subtipo são separados por uma barra, como em: 'Content-Type: video/mpeg'. Desde então, centenas de subtipos foram acrescentados, junto com outro tipo. Outras entradas estão sendo acrescentadas o tempo todo à medida que novos tipos de conteúdo são desenvolvidos. A lista de tipos e subtipos especificados é mantida on-line pela IANA em www.iana.org/assignments/media-types.

Os tipos, juntamente com exemplos de subtipos mais utilizados, aparecem na Tabela 7.6. Vamos analisá-los rapidamente, começando com *text*. A combinação *text/plain* é para mensagens comuns, que podem ser exibidas conforme recebidas, sem nenhuma codificação ou processamento adicional. Essa opção permite que mensagens comuns sejam transportadas em MIME com apenas alguns cabeçalhos extras. O subtipo *text/html* foi acrescentado quando a Web se tornou popular (na RFC 2854) para permitir que as páginas Web fossem enviadas no e-mail da RFC 822. Um subtipo para a eXtensible Markup Language, *text/xml*, é definido na RFC 3023. Os documentos XML proliferaram com o desenvolvimento da Web. Estudaremos HTML e XML na Seção 7.3.

O próximo tipo MIME é *image*, usado para transmitir fotografias. Atualmente, são utilizados muitos formatos para armazenar e transmitir imagens, com e sem compressão. Vários deles, incluindo GIF, JPEG e TIFF, estão incorporados em quase todos os navegadores, mas também existem muitos outros formatos e subtipos correspondentes.

Os tipos *audio* e *video* se destinam a som e imagens em movimento, respectivamente. Observe que a opção *video* inclui apenas as informações visuais, e não a trilha sonora. Se um filme com som tiver de ser transmitido, as partes de vídeo e áudio talvez tenham de ser transmitidas separadamente, dependendo do sistema de codificação utilizado. O primeiro formato de vídeo definido foi criado pelo grupo Moving Pictures Experts Group (MPEG), mas desde então foram acrescentados outros formatos. Além de *audio/basic*, foi incluído na RFC 3003 um novo tipo de áudio, *audio/*

Tipo	Subtipos de exemplo	Descrição
text	plain, html, xml, css	Texto em vários formatos
image	gif, jpeg, tiff	Imagens
audio	basic, mpeg, mp4	Sons
video	mpeg, mp4, quicktime	Filmes
model	vrml	Modelo 3D
application	octect-stream, pdf, javascript, zip	Dados produzidos por aplicações
message	http, rfc822	Mensagem encapsulada
multipart	mixed, alternative, parallel, digest	Combinação de vários tipos

Tabela 7.6 | Tipos de conteúdo MIME e exemplos de subtipos.

mpeg, para permitir que as pessoas enviem arquivos de áudio MP3 por correio eletrônico. Os tipos *video/mp4* e *audio/mp4* sinalizam dados de vídeo e áudio que são armazenados no formato MPEG 4, mais novo.

O tipo *model* foi acrescentado após os outros tipos de conteúdo. Ele serve para descrever dados de modelo 3D. Porém, até o momento ele não foi muito utilizado.

O tipo *application* é um depósito para formatos não cobertos por um dos outros tipos e que exigem uma aplicação para interpretar os dados. Listamos os subtipos *pdf, javascript* e *zip* como exemplos para documentos PDF, programas em JavaScript e arquivos Zip, respectivamente. Os agentes do usuário que recebem esse conteúdo utilizam uma biblioteca de terceiros ou um programa externo para exibir o conteúdo; a exibição pode ou não estar integrada ao agente do usuário.

Usando tipos MIME, os agentes do usuário ganham a facilidade de extensão para lidar com novos tipos de conteúdo de aplicação à medida que são desenvolvidos. Esse é um benefício significativo. Por outro lado, muitas das novas formas de conteúdo são executadas ou interpretadas pelas aplicações, o que apresenta alguns perigos. Sem dúvida, rodar um programa executável qualquer que chegou pelo sistema de e-mail vindo de 'amigos' traz um risco à segurança. O programa pode realizar todos os tipos de danos às partes do computador às quais tem acesso, especialmente se puder ler e gravar arquivos e usar a rede. De modo menos óbvio, formatos de documento podem impor os mesmos riscos. Isso porque formatos como PDF são linguagens de programação poderosas e disfarçadas. Embora sejam interpretadas e restritas em escopo, bugs no interpretador em geral permitem que documentos nocivos escapem às restrições.

Além desses exemplos, há muitos outros subtipos, pois existem muito mais aplicações. Como uma saída a ser usada quando nenhum outro subtipo conhecido é mais adequado, o *octet-stream* indica uma sequência de bytes não interpretados. Ao receber essa sequência, é provável que um agente do usuário o apresente na tela, sugerindo que ele seja copiado para um arquivo. O processamento a seguir fica a critério do usuário, que provavelmente conhece o tipo do conteúdo.

Os dois últimos tipos são úteis para compor e manipular as próprias mensagens. O tipo *message* permite que uma mensagem seja totalmente encapsulada em outra. Esse esquema é útil para, por exemplo, encaminhar mensagens de e-mail. Quando uma mensagem RFC 822 completa é encapsulada em uma mensagem externa, deve-se utilizar o subtipo *rfc822*. De modo semelhante, é comum que documentos HTML sejam encapsulados. O subtipo *partial* permite a divisão de uma mensagem encapsulada em partes e o envio dessas partes separadas (por exemplo, se a mensagem encapsulada for longa demais). Os parâmetros tornam possível montar novamente todas as partes em ordem correta no destino.

O último tipo é *multipart*, que permite a uma mensagem conter mais de uma parte, com o início e o fim de cada parte claramente delimitados. O subtipo *mixed* aceita que cada parte seja diferente, sem a imposição de uma estrutura adicional. Muitos programas de e-mail autorizam o usuário a incluir um ou mais anexos em uma mensagem de texto. Esses anexos são enviados usando-se o tipo *multipart*.

Em contraste com *mixed*, o subtipo *alternative* permite que a mesma mensagem seja incluída várias vezes, mas expressa em dois ou mais meios. Por exemplo, uma mensagem poderia ser enviada em código ASCII simples, em HTML e em PDF. Um agente do usuário projetado de forma adequada que recebesse essa mensagem poderia exibi-la de acordo com as preferências do usuário. A segunda opção seria HTML. Se nenhuma dessas opções fosse possível, o arquivo ASCII puro seria exibido. As partes seriam ordenadas da mais simples para a mais complexa, para que os destinatários com agentes do usuário anteriores ao lançamento do MIME pudessem compreender a mensagem (por exemplo, até mesmo um usuário com sistema anterior ao MIME poderia ler um texto ASCII simples).

O subtipo *alternative* também pode ser usado para vários idiomas. Nesse contexto, a Pedra de Roseta pode ser considerada uma antiga mensagem *multipart/alternative*.

Dos outros dois subtipos, o *parallel* é usado quando todas as partes devem ser 'vistas' simultaneamente. Por exemplo, com frequência os filmes têm um canal de áudio e um canal de vídeo. Os filmes serão mais efetivos se esses dois canais forem reproduzidos em paralelo, e não sucessivamente. O subtipo *digest* é usado quando muitas mensagens são compactadas em uma mensagem composta. Por exemplo, alguns grupos de discussão da Internet colecionam mensagens de assinantes e as enviam como uma única mensagem *multipart/digest*.

Um exemplo de como os tipos MIME podem ser usados para mensagens de e-mail é uma mensagem multimídia, como no Quadro 7.2. Nele, uma mensagem de aniversário é transmitida em formas alternativas como HTML e como um arquivo de áudio. Supondo que o receptor tenha capacidade de áudio, o agente do usuário tocará o arquivo de som. Nesse exemplo, o som é transportado por referência como um subtipo *message/external-body*, de modo que primeiro o agente do usuário precisa buscar o arquivo de som *birthday.snd* usando FTP. Se o agente do usuário não tiver capacidade de áudio, a letra será exibida na tela em silêncio total. As duas partes são delimitadas por dois hifens seguidos por uma string (gerada pelo software) especificada no parâmetro *boundary*.

Observe que o cabeçalho *Content-Type* ocorre em três posições nesse exemplo. No nível mais alto, ele indica que a mensagem tem várias partes. Dentro de cada parte, são indicados seu tipo e subtipo. Por fim, dentro do corpo da segunda parte, esse cabeçalho é necessário para informar ao agente do usuário que tipo de arquivo externo será obtido. Para indicar essa pequena diferença em termos de uso,

```
From: alice@cs.washington.edu
To: bob@ee.uwa.edu.au
MIME-Version: 1.0
Message-Id: <0704760941.AA00747@cs.washington.edu>
Content-Type: multipart/alternative; boundary = qwertyuiopasdfghjklzxcvbnm
Subject: Earth orbits sun integral number of times

This is the preamble. The user agent ignores it. Have a nice day.

--qwertyuiopasdfghjklzxcvbnm
Content-Type: text/html

<p>Happy birthday to you<br>
Happy birthday to you<br>
Happy birthday dear <b> Bob </b><br>
Happy birthday to you</p>

--qwertyuiopasdfghjklzxcvbnm
Content-Type: message/external-body;
    access-type = "anon-ftp";
    site = "bicycle.cs.washington.edu";
    directory = "pub";
    name = "birthday.snd"

content-type: audio/basic
content-transfer-encoding: base64
--qwertyuiopasdfghjklzxcvbnm--
```

Quadro 7.2 | Uma mensagem em várias partes contendo diretivas HTML e áudio.

empregamos letras minúsculas, apesar de nenhum cabeçalho fazer diferença entre maiúsculas e minúsculas. O cabeçalho *Content-Transfer-Encoding* também é necessário para qualquer corpo externo que não esteja codificado como ASCII de sete bits.

7.2.4 Transferência de mensagens

Agora que descrevemos os agentes do usuário e as mensagens de e-mail, estamos prontos para examinar como os agentes de transferência de mensagens repassam mensagens do remetente ao destinatário. A transferência de e-mail é feita com o protocolo SMTP.

A maneira mais simples de mover mensagens é estabelecer uma conexão de transporte entre a máquina de origem e a de destino e, em seguida, transferir a mensagem. É assim que o STMP funcionava originalmente. Com o passar do tempo, porém, dois usos diversos do SMTP foram estabelecidos. O primeiro uso é o **envio de correio**, a etapa 1 na arquitetura de e-mail da Figura 7.4. Esse é o meio pelo qual os agentes do usuário enviam mensagens para o sistema de e-mail para entrega. O segundo uso é para transferir mensagens entre agentes de transferência de mensagem (etapa 2 na Figura 7.4). Essa sequência entrega o e-mail do agente de transferência de mensagem emissor ao receptor, em um hop. A entrega final é realizada com diferentes protocolos, os quais descreveremos na próxima seção.

Nesta seção, vamos descrever os fundamentos do protocolo SMTP e seu mecanismo de extensão. Depois, discutiremos como ele é usado de modo diferente para envio de correio e transferência de mensagem.

SMTP (Simple Mail Transfer Protocol) e extensões

Dentro da Internet, as mensagens de correio eletrônico são entregues quando a máquina de origem estabelece uma conexão TCP com a porta 25 da máquina de destino. Um servidor de correio que se comunica em **SMTP (Simple Mail Transfer Protocol)** permanece na escuta nessa porta. Esse servidor aceita as conexões recebidas, sujeitas a algumas verificações de segurança, e também mensagens para entrega. Se uma mensagem não puder ser entregue, um relatório de erros contendo a primeira parte da mensagem não entregue será retornado ao remetente.

O SMTP é um protocolo ASCII muito simples. Isso não é um ponto fraco, mas um recurso. O uso de texto ASCII torna os protocolos fáceis de desenvolver, testar e depurar. Eles podem ser testados enviando comandos manualmente, e os registros das mensagens são fáceis de ler. A maioria dos protocolos da Internet em nível de aplicação agora trabalha dessa maneira (por exemplo, HTTP).

Vamos examinar uma transferência de mensagem simples entre servidores de correio que entregam uma mensagem. Após estabelecer a conexão TCP com a porta 25, a máquina de transmissão, operando como cliente, espera que a máquina de recepção, operando como servidor, comunique-se primeiro. O servidor começa enviando uma linha de texto que fornece sua identidade e informa que está preparado para receber mensagens. Caso não esteja, o cliente encerrará a conexão e tentará outra vez mais tarde.

Se o servidor estiver disposto a receber, o cliente anunciará de quem veio a mensagem e para quem ela está indo. Se esse receptor existir no local de destino, o servidor dará ao cliente o sinal para enviar a mensagem. Em seguida, o cliente a enviará e o servidor a confirmará. Não são necessários checksums, porque o TCP fornece um fluxo de bytes confiável. Se houver mais mensagens, elas serão enviadas. Quando todas as mensagens tiverem sido trocadas em ambos os sentidos, a conexão será encerrada. Um exemplo do diálogo necessário para o envio da mensagem ilustrada no Quadro 7.2, incluindo os códigos numéricos utilizados pelo SMTP, é mostrado no Quadro 7.3. As linhas enviadas pelo cliente (o transmissor) são marcadas como *C:*. As linhas enviadas pelo servidor (o receptor) são marcadas como *S:*.

O primeiro comando enviado pelo cliente é *HELO*. Entre as várias abreviaturas de quatro caracteres para *HELLO*, essa tem muito mais vantagens do que sua maior concorrente. A razão para que todos os comandos tivessem quatro caracteres se perdeu no tempo.

No Quadro 7.3, a mensagem é enviada para um único destinatário; portanto, apenas um comando RCPT é usado. Esses comandos são usados para enviar uma única mensagem a vários destinatários. Cada um deles é confirmado ou rejeitado individualmente. Ainda que alguns destinatários sejam rejeitados (por não existirem no destino), a mensagem poderá ser enviada aos outros.

Por fim, apesar de a sintaxe dos comandos de quatro caracteres enviados pelo cliente ser especificada de forma bastante rígida, a sintaxe das respostas é mais flexível. Apenas o código numérico é importante. Cada implementação pode incluir uma string qualquer depois do código.

O SMTP básico funciona bem, mas é limitado em vários aspectos. Ele não inclui autenticação. Isso significa que o comando *FROM* no exemplo poderia dar qualquer endereço de transmissor que ele quisesse. Isso é muito útil para enviar spam. Outra limitação é que o SMTP transfere mensagens ASCII, e não dados binários. É por isso que a codificação de transferência de conteúdo MIME base64 foi necessária. Porém, com essa codificação, a transmissão de correio usa a largura de banda de modo ineficaz, o que é um problema para mensagens grandes. Uma terceira limitação é que o SMTP envia mensagens às claras. Ele não tem criptografia para fornecer uma forma de privacidade contra olhares curiosos.

Para permitir que esses e muitos outros problemas relacionados ao processamento da mensagem sejam resolvidos, o SMTP foi revisado para ter um mecanismo de extensão. Esse mecanismo é uma parte obrigatória do padrão RFC 5321. O uso do SMTP com extensões é chamado **ESMTP** (**Extended SMTP**).

Os clientes que desejam usar uma extensão enviam uma mensagem *EHLO* em vez de *HELO* inicialmente. Se esta for rejeitada, o servidor é um SMTP comum, e o cliente deverá prosseguir pelo modo normal. Se o *EHLO* for aceito, o servidor responde com as extensões que ele aceita. O cliente pode então usar qualquer uma dessas extensões. Várias extensões comuns aparecem na Tabela 7.7. A figura mostra a palavra-chave usada no mecanismo de extensão, com uma descrição da nova funcionalidade. Não examinaremos essas extensões com mais detalhes.

Para entender melhor como o SMTP e alguns dos outros protocolos descritos neste capítulo funcionam, experimente-os. Em todos os casos, vá primeiro até um equipamento conectado à Internet. Em um sistema UNIX (ou Linux), digite em um shell:

```
telnet mail.isp.com 25
```

Palavra-chave	Descrição
AUTH	Autenticação do cliente
BINARYMIME	Servidor aceita mensagens binárias
CHUNKING	Servidor aceita mensagens grandes em pedaços
SIZE	Verificar tamanho da mensagem antes de tentar enviar
STARTTLS	Passar para transporte seguro (TLS; ver Capítulo 8)
UTF8SMTP	Endereços internacionalizados

Tabela 7.7 | Algumas extensões SMTP.

```
                S: 220 ee.uwa.edu.au SMTP service ready
C: HELO abcd.com
                S: 250 cs.washington.edu says hello to ee.uwa.edu.au
C: MAIL FROM: <alice@cs.washington.edu>
                S: 250 sender ok
C: RCPT TO: <bob@ee.uwa.edu.au>
                S: 250 recipient ok
C: DATA
                S: 354 Send mail; end with "." on a line by itself
C: From: alice@cs.washington.edu
C: To: bob@ee.uwa.edu.au
C: MIME-Version: 1.0
C: Message-Id: <0704760941.AA00747@ee.uwa.edu.au>
C: Content-Type: multipart/alternative; boundary = qwertyuiopasdfghjklzxcvbnm
C: Subject: Earth orbits sun integral number of times
C:
C: This is the preamble. The user agent ignores it. Have a nice day.
C:
C: --qwertyuiopasdfghjklzxcvbnm
C: Content-Type: text/html
C:
C: <p>Happy birthday to you
C: Happy birthday to you
C: Happy birthday dear <bold> Bob </bold>
C: Happy birthday to you
C:
C: --qwertyuiopasdfghjklzxcvbnm
C: Content-Type: message/external-body;
C:      access-type = "anon-ftp";
C:      site = "bicycle.cs.washington.edu";
C:      directory = "pub";
C:      name = "birthday.snd"
C:
C: content-type: audio/basic
C: content-transfer-encoding: base64
C: --qwertyuiopasdfghjklzxcvbnm
C: .
                S: 250 message accepted
C: QUIT
                S: 221 ee.uwa.edu.au closing connection
```

Quadro 7.3 | Transferência de uma mensagem de alice@cs.washington.edu para bob@ee.uwa.edu.au.

substituindo mail.isp.com pelo nome DNS do servidor de correio do seu provedor. Em um sistema Windows, clique em Iniciar, depois em Executar e, em seguida, digite o comando na caixa de diálogo. Esse comando estabelecerá uma conexão telnet (isto é, TCP) para a porta 25 nessa máquina. A porta 25 é a porta SMTP (veja na Tabela 6.3 algumas portas comuns). Provavelmente, você obterá uma resposta semelhante a esta:

Trying 192.30.200.66...

Connected to mail.isp.com

Escape character is ’]’.

220 mail.isp.com Smail #74 ready at Thu, 25 Sept 2002 13:26 +0200

As três primeiras linhas são do telnet, informando-lhe o que está fazendo. A última linha é do servidor SMTP na máquina remota, anunciando sua disposição para se comunicar com você e aceitar mensagens de correio eletrônico. Para descobrir que comandos ele aceita, digite:

HELP

Desse ponto em diante, é possível uma sequência de comandos como a da Tabela 7.7, se o servidor estiver disposto a aceitar seu e-mail.

Envio de correio

No início, os agentes do usuário funcionavam no mesmo computador do agente de transferência de mensagem. Nesse esquema, tudo o que é necessário para enviar uma mensagem é que o agente do usuário fale com o servidor de correio local, usando o diálogo que acabamos de descrever. Porém, esse esquema não é mais o caso comum.

Os agentes do usuário normalmente são executados em notebooks, PCs domésticos e telefones móveis. Eles nem sempre estão conectados à Internet. Os agentes de transferência de correio são executados no ISP e nos servidores da empresa. Eles sempre estão conectados à Internet. Essa diferença significa que um agente do usuário em Recife pode precisar entrar em contato com seu servidor de correio normal em Curitiba para enviar uma mensagem de correio, pois o usuário está viajando.

Por si só, essa comunicação remota não causa problema algum. É exatamente para isso que os protocolos TCP/IP são preparados a dar esse suporte. Porém, um ISP ou uma empresa em geral não desejam que qualquer usuário remoto possa submeter mensagens ao seu servidor de correio para ser entregue em outro lugar. O ISP ou a empresa não estão executando o servidor como um serviço público. Além disso, esse tipo de **repasse de correio aberto** atrai os spammers. Isso porque ele oferece um modo de se passar pelo remetente original e, portanto, tornar a mensagem mais difícil de ser identificada como spam.

Com essas considerações, o SMTP normalmente é usado para envio de correio com a extensão *AUTH*. Essa extensão permite que o servidor verifique as credenciais (nome de usuário e senha) do cliente para confirmar que o servidor deve estar oferecendo serviço de correio.

Existem várias outras diferenças no modo como o SMTP é usado para envio de correio. Por exemplo, a porta 587 é usada em preferência à porta 25 e o servidor SMTP pode verificar e corrigir o formato das mensagens enviadas pelo agente do usuário. Para obter mais informações sobre o uso restrito do SMTP para envio de correio, consulte a RFC 4409.

Transferência de mensagem

Quando o agente de transferência de correio de saída recebe uma mensagem do agente do usuário, ele a entrega ao agente de transferência de correio de entrada usando SMTP. Para fazer isso, o transmissor usa o endereço de destino. Considere a mensagem no Quadro 7.3, endereçada para bob@ee.uwa.edu.au. Para qual servidor de correio a mensagem deve ser entregue?

Para determinar o servidor de correio correto para contatar, o DNS é consultado. Na seção anterior, descrevemos como o DNS contém vários tipos de registros, incluindo o registro *MX*, ou trocador de mensagens eletrônicas. Nesse caso, uma consulta do DNS é feita aos registros *MX* do domínio ee.uwa.edu.au. Essa consulta retorna uma lista ordenada dos nomes e endereços IP de um ou mais servidores de correio.

O agente de transferência de correio de saída, em seguida, faz uma conexão TCP com a porta 25 do endereço IP do servidor de correio para alcançar o agente de transferência de correio de entrada, e usa SMTP para repassar a mensagem. O agente de transferência de correio de entrada colocará o e-mail para o usuário *bob* na caixa de correio correta para Bob, para lê-lo posteriormente. Essa etapa de entrega local pode envolver a movimentação da mensagem entre computadores, se houver uma infraestrutura de correio.

Com esse processo de entrega, o e-mail trafega do agente de transferência de correio inicial para o final em um único hop. Não existem servidores intermediários no estágio de transferência de mensagem. Porém, é possível que esse processo de entrega ocorra várias vezes. Um exemplo que já descrevemos é quando um agente de transferência de mensagem implementa uma lista de correspondência. Nesse caso, uma mensagem é recebida na lista. Depois, ela é expandida como uma mensagem para cada membro da lista, que é enviada para os endereços individuais dos membros.

Como outro exemplo de repasse, Bob pode ter se formado no MIT e também pode ser alcançado pelo endereço bob@alum.mit.edu. Em vez de ler o e-mail em várias contas, Bob pode fazer com que a mensagem enviada para esse endereço seja encaminhada para bob@ee.uwa.edu. Nesse caso, o e-mail enviado para bob@alum.mit.edu terá duas entregas. Primeiro, para o servidor de correio para alum.mit.edu. Depois, para o servidor de correio para ee.uwa.edu.au. Cada uma dessas pernas é uma entrega completa e separada, trata-se dos agentes de transferência de correio.

Outra consideração atual é o spam. Nove entre dez mensagens enviadas hoje são spams (McAfee, 2010). Poucas pessoas ainda desejam spam, mas isso é difícil de evitar, pois ele é disfarçado como correio normal. Antes de aceitar uma mensagem, verificações adicionais podem ser enviadas para reduzir os riscos de spam. A mensagem para Bob foi enviada de alice@cs.washington.edu. O agente de transferência de correio de entrada pode consultar o agente de transferência de correio de saída no DNS. Isso permite que

ele verifique se o endereço IP do outro lado da conexão TCP corresponde ao nome do DNS. Geralmente, o agente receptor pode pesquisar o domínio transmissor no DNS para ver se ele tem uma política de envio de correio. Essa informação normalmente é dada nos registros *TXT* e *SPF*. Ela pode indicar que outras verificações podem ser feitas. Por exemplo, o e-mail enviado de cs.washington.edu pode sempre ser enviado do host june.cs.washington.edu. Se o agente de transferência de correio de saída não for *june*, existe um problema.

Se qualquer uma dessas verificações falhar, o e-mail provavelmente estará sendo forjado com um endereço de envio falso. Nesse caso, ele é descartado. Porém, passar nessas verificações não significa que o e-mail não seja spam. As verificações simplesmente garantem que o e-mail parece estar vindo da região da rede de onde declara ter vindo. A ideia é que os spammers devem ser forçados a usar o endereço de envio correto quando enviarem e-mail. Isso torna o spam mais fácil de ser reconhecido e excluído quando for indesejado.

7.2.5 Entrega final

Nossa mensagem de e-mail está quase entregue. Ela chegou à caixa de correio de Bob. Tudo o que resta é transferir uma cópia da mensagem para o agente do usuário de Bob para exibição. Essa é a etapa 3 na arquitetura da Figura 7.4. Essa tarefa era simples no início da Internet, quando o agente do usuário e o agente de transferência de correio eram executados na mesma máquina como processos diferentes. O agente de transferência de correio simplesmente escrevia novas mensagens no final do arquivo da caixa de correio, e o agente do usuário simplesmente verificava se havia novo e-mail na caixa de correio.

Atualmente, o agente do usuário em um PC, notebook ou dispositivo móvel provavelmente estará em uma máquina diferente do ISP ou do servidor de correio da empresa. Os usuários querem poder acessar seu e-mail remotamente, de onde quer que estiverem. Eles querem acessar o e-mail do trabalho, dos seus PCs em casa, dos seus notebooks quando estiverem viajando e de LAN houses quando estiverem em, digamos assim, férias. Eles também querem poder trabalhar off-line, depois reconectar para receber o e-mail que chega e enviar e-mail. Além do mais, cada usuário pode executar vários agentes do usuário, dependendo de qual computador é conveniente para usar no momento. Vários agentes do usuário podem ainda estar rodando ao mesmo tempo.

Nesse esquema, o trabalho do agente do usuário é apresentar uma visão do conteúdo da caixa de correio e permitir que esta seja manipulada remotamente. Vários protocolos podem ser usados para essa finalidade, mas o SMTP é um deles. O SMTP é um protocolo do tipo push. Ele captura uma mensagem e se conecta a um servidor remoto para transferi-la. A remessa final não pode ser obtida dessa maneira, porque a caixa de correio precisa continuar a ser armazenada no agente de transferência de correio e o agente do usuário pode não estar conectado à Internet no momento em que o SMTP tenta repassar as mensagens.

IMAP – Internet Message Access Protocol

Um dos principais protocolos usados para remessa final é o **IMAP** (**Internet Message Access Protocol**). A versão 4 do protocolo é definida na RFC 3501. Para usar o IMAP, o servidor de correio executa um servidor IMAP que escuta na porta 143. O agente do usuário executa um cliente IMAP. O cliente se conecta ao servidor e começa a emitir comandos a partir daqueles listados na Tabela 7.8.

Primeiro, o cliente iniciará um transporte seguro, se tiver que ser usado (a fim de manter as mensagens e os comandos confidenciais), e depois fará um login ou se autenticará de alguma maneira no servidor. Uma vez autenticado, haverá muitos comandos para listar pastas e mensagens, buscar mensagens ou ainda partes delas, marcá-las com flags para posterior exclusão e organizá-las em pastas. Para evitar confusão, por favor, observe que usamos o termo 'pasta' aqui para ser coerentes com o restante do material nesta seção, em que um usuário tem uma única caixa de correio composta de várias pastas. Porém, na especificação IMAP, o termo *caixa de correio* (*mailbox*) é usado em seu lugar. Um usuário, portanto, tem muitas caixas de correio IMAP, cada uma normalmente apresentada ao usuário como uma pasta.

O IMAP também possui muitos outros recursos. Ele tem a capacidade de endereçar e-mail não por número de mensagem, mas usando atributos (por exemplo, dê-me a primeira mensagem de Alice). As consultas podem ser realizadas no servidor para encontrar as mensagens que satisfazem a certos critérios, de modo que apenas essas mensagens sejam capturadas pelo cliente.

O IMAP é uma melhoria em relação a um protocolo de entrega final mais antigo, o **POP3** (**Post Office Protocol**, **version 3**), que é especificado na RFC 1939. O POP3 é um protocolo mais simples, mas admite menos recursos e é menos seguro no uso diário. O e-mail normalmente é baixado para o computador do agente do usuário, em vez de permanecer no servidor de correio. Isso facilita a vida no servidor, mas é mais difícil para o usuário. Não é fácil ler o e-mail em vários computadores, além do que, se o computador do agente do usuário quebrar, todo o e-mail pode se perder permanentemente. Apesar disso, você ainda encontrará o POP3 sendo usado.

Protocolos fechados também podem ser usados, pois o protocolo atua entre um servidor de correio e o agente do usuário, que pode ser fornecido pela mesma empresa. O Microsoft Exchange é um sistema de correio com um protocolo fechado.

Comando	Descrição
CAPABILITY	Lista capacidades do servidor
STARTTLS	Inicia o transporte seguro (TLS; ver Capítulo 8)
LOGIN	Login no servidor
AUTHENTICATE	Login com outro método
SELECT	Seleciona uma pasta
EXAMINE	Seleciona uma pasta apenas de leitura
CREATE	Cria uma pasta
DELETE	Exclui uma pasta
RENAME	Renomeia uma pasta
SUBSCRIBE	Acrescenta pasta do conjunto ativo
UNSUBSCRIBE	Remove pasta do conjunto ativo
LIST	Lista as pastas disponíveis
LSUB	Lista as pastas ativas
STATUS	Captura o status de uma pasta
APPEND	Acrescenta uma mensagem a uma pasta
CHECK	Captura um ponto de verificação de uma pasta
FETCH	Captura mensagens de uma pasta
SEARCH	Localiza mensagens em uma pasta
STORE	Altera flags de mensagem
COPY	Faz uma cópia de uma mensagem em uma pasta
EXPUNGE	Remove mensagens marcadas para exclusão
UID	Emite comandos usando identificadores exclusivos
NOOP	Não faz nada
CLOSE	Remove mensagens marcadas e fecha pasta
LOGOUT	Efetua o logout e fecha a conexão

Tabela 7.8 | Comandos IMAP (versão 4).

Webmail

Uma alternativa cada vez mais popular ao IMAP e ao SMTP para fornecer serviço de e-mail é usar a Web como interface para enviar e receber e-mail. Os serviços de **Webmail** mais usados são Google Gmail, Microsoft Hotmail e Yahoo! Mail. Webmail é um exemplo de software (neste caso, um agente do usuário de correio) que é fornecido como um serviço de uso da Web.

Nessa arquitetura, o provedor executa serviços de correio normalmente para aceitar mensagens para usuários com SMTP na porta 25. Porém, o agente do usuário é diferente. Em vez de ser um programa isolado, ele é uma interface do usuário fornecida por páginas Web. Isso significa que os usuários podem usar qualquer navegador que quiserem para acessar seu e-mail e enviar novas mensagens.

Ainda não estudamos a Web, mas apresentaremos a seguir uma descrição rápida, que você poderá consultar quando for preciso. Quando o usuário vai até a página Web de e-mail do provedor, é apresentado um formulário no qual ele deve informar um nome de login e sua senha. O nome de login e a senha são enviados ao servidor, que então os valida. Se o login tiver sucesso, o servidor encontra a caixa de correio do usuário e monta uma página Web listando o conteúdo da caixa de correio na hora. A página Web é então enviada para ser exibida pelo navegador.

Muitos dos itens na página que mostram a caixa de correio são clicáveis, de modo que as mensagens podem ser lidas, excluídas etc. Para tornar a interface responsiva, as páginas Web normalmente incluirão programas em JavaScript. Esses programas são executados localmente no

cliente em resposta a eventos locais (por exemplo, cliques do mouse) e também podem baixar e atualizar mensagens em segundo plano, para preparar a próxima mensagem para exibição ou uma nova mensagem para envio. Nesse modelo, o envio de correio acontece usando os protocolos normais da Web, postando dados para um URL. O servidor Web cuida da injeção de mensagens no sistema de entrega de correio tradicional, que descrevemos anteriormente. Por segurança, os protocolos-padrão da Web também podem ser usados. Esses protocolos cuidam da criptografia de páginas Web, e não se o conteúdo da página Web é uma mensagem de e-mail.

7.3 A World Wide Web

A Web, como a World Wide Web é conhecida, é uma estrutura arquitetônica que permite o acesso a documentos vinculados espalhados por milhões de máquinas na Internet. Em dez anos, ela deixou de ser um meio de distribuição de dados sobre física de alta energia na Suíça para se tornar a aplicação que milhões de pessoas consideram ser 'A Internet'. Sua enorme popularidade se deve à interface gráfica colorida, de fácil utilização para principiantes. Além disso, ela oferece uma imensa variedade de informações sobre quase todos os assuntos imagináveis, desde aborígines até zoologia.

A Web começou em 1989 no CERN, o European Center for Nuclear Research. A ideia inicial foi ajudar grandes equipes, geralmente com membros em meia dúzia de países ou mais e fusos horários diferentes, a colaborar usando uma coleção de relatórios, plantas, desenhos, fotos e outros documentos que mudam constantemente, produzidos por experimentos em física de partículas. A proposta para uma teia de documentos interligados veio do físico do CERN Tim Berners-Lee. O primeiro protótipo (baseado em texto) entrou em operação 18 meses depois. Uma demonstração pública na conferência Hypertext '91 chamou a atenção de outros pesquisadores, levando Marc Andreessen, da Universidade de Illinois, a desenvolver o primeiro navegador gráfico. Ele foi chamado Mosaic e lançado em fevereiro de 1993.

O restante, como dizem, agora é história. O Mosaic foi tão popular que um ano depois Andreessen saiu para formar uma empresa, a Netscape Communications Corp., cujo objetivo era desenvolver software para a Web. Pelos três anos seguintes, o Netscape Navigator e o Internet Explorer, da Microsoft, entraram em uma 'batalha de navegadores', cada um tentando capturar uma fatia maior do novo mercado, acrescentando freneticamente mais recursos (e, portanto, mais bugs) que a outra.

No decorrer das décadas de 90 e 2000, sites e páginas Web, como o conteúdo Web é chamado, cresceram exponencialmente até que houvesse milhões de sites e bilhões de páginas. Um pequeno número desses sites se tornou tremendamente popular. Esses sites e as empresas por trás deles definem em grande parte a Web conforme as pessoas a veem hoje. Alguns exemplos são: uma livraria (Amazon, iniciada em 1994, capitalização de mercado de US$ 50 bilhões), um mercado de leilões (eBay, 1995, US$ 30 bilhões), busca (Google, 1998, US$ 150 bilhões) e rede social (Facebook, 2004, empresa privada com valor de mais de US$ 15 bilhões). O período da década de 2000, em que muitas empresas da Web passaram a valer centenas de milhões de dólares da noite para o dia, apenas para praticamente falir no dia seguinte, quando saíam de moda, tem até mesmo um nome específico. Ele se chama **era ponto com**. Novas ideias ainda estão chegando à Web. Muitas delas vêm de estudantes. Por exemplo, Mark Zuckerberg era um aluno de Harvard quando iniciou o Facebook, e Sergey Brin e Larry Page eram alunos de Stanford quando iniciaram o Google. Talvez você seja o próximo a surgir com uma ideia brilhante.

Em 1994, o CERN e o MIT assinaram um acordo criando o **W3C (World Wide Web Consortium)**, uma organização voltada para o desenvolvimento da Web, padronização de protocolos e para o incentivo da interoperabilidade entre os sites. Berners-Lee tornou-se o diretor. Desde então, centenas de universidades e empresas juntaram-se ao consórcio. Embora existam agora mais livros sobre a Web do que se possa imaginar, o melhor lugar para obter informações atualizadas é (naturalmente) a própria Web. A home page do consórcio está no endereço www.w3.org. Os leitores interessados são levados por links a páginas que englobam todos os numerosos documentos e atividades do consórcio.

7.3.1 Visão geral da arquitetura

Do ponto de vista dos usuários, a Web é uma vasta coleção mundial de documentos, geralmente chamados **páginas Web**, ou apenas **páginas**. Cada página pode conter links (vínculos) para outras páginas em qualquer lugar do mundo. Os usuários podem seguir um link (por exemplo, dando um clique sobre ele) que os levará até a página indicada. Esse processo pode ser repetido indefinidamente. A ideia de fazer uma página apontar para outra, agora chamada **hipertexto**, foi criada por um visionário professor de engenharia elétrica do MIT, Vannevar Bush, em 1945 (Bush, 1945), bem antes da criação da Internet. De fato, isso foi antes da existência dos computadores comerciais, embora várias universidades tenham produzido protótipos primitivos que preenchiam grandes salas e tinham menos potência do que uma calculadora de bolso moderna.

As páginas geralmente são visualizadas com o auxílio de um programa denominado **navegador**. Firefox, Internet Explorer e Chrome são exemplos de navegadores conhecidos. O navegador busca a página solicitada, interpreta seu conteúdo e exibe a página, formatada de modo apropriado, na tela do computador. O conteúdo em si pode ser uma mistura de texto, imagens e comandos de formatação, na forma

de um documento tradicional, ou outras formas de conteúdo, como vídeo ou programas que produzem uma interface gráfica com a qual os usuários podem interagir.

Um exemplo de como isso é feito encontra-se na Figura 7.7. Essa é a página para o departamento de Ciência da Computação e Engenharia na Universidade de Washington. Essa página mostra elementos de texto e gráficos (que normalmente são muito pequenos para ser lidos). Algumas partes da página estão associadas a links para outras páginas. Um pedaço de texto, ícone, imagem e assim por diante, associados a outra página, são chamados **hiperlink**. Seguir um link é simplesmente um modo de dizer ao navegador para buscar outra página. Nos primeiros dias da Web, os links eram destacados com sublinhado e texto colorido, para que pudessem ser destacados. Hoje, os criadores de páginas Web têm meios de controlar a aparência das regiões interligadas, de modo que um link pode se parecer com um ícone ou mudar sua aparência quando o mouse passa sobre ele. Os criadores da página são responsáveis por tornar os links visualmente distintos, para fornecer uma interface atraente.

Os alunos no departamento podem descobrir mais seguindo um link para uma página com informações especialmente para eles. O link é acessado com um clique na área circulada. O navegador, então, busca a nova página e a exibe, como mostramos parcialmente no canto inferior esquerdo da Figura 7.7. Dúzias de outras páginas estão ligadas à primeira página além deste exemplo. Cada outra página pode ser composta de conteúdo na mesma máquina (ou máquinas) que a primeira, ou em máquinas no outro lado do mundo. O usuário não sabe disso. A busca de página é feita pelo navegador, sem nenhuma ajuda do usuário. Assim, a movimentação entre as máquinas enquanto o conteúdo é exibido é transparente.

O modelo básico por trás da exibição de páginas também é apresentado na Figura 7.7. O navegador está exibindo uma página Web na máquina do cliente. Cada página é capturada enviando uma solicitação a um ou mais servidores, que respondem com o conteúdo da página. O protocolo de solicitação-resposta para buscar páginas é simples, baseado em texto, que roda sobre TCP, assim como no caso do SMTP. Ele é chamado **HTTP** (**HyperText Transfer Protocol**). O conteúdo pode simplesmente ser um documento que é lido de um disco, ou então o resultado de uma consulta de banco de dados e execução de programa. Quanto à página, ela pode ser uma **página estática** se apresentar o mesmo documento toda vez que for exibida. Ao contrário, se ela foi gerada sob demanda por um programa ou se contém um programa, é chamada **página dinâmica**.

Uma página dinâmica pode se apresentar de forma diferente toda vez que for exibida. Por exemplo, a página inicial de uma loja virtual pode ser diferente para cada visitante. Se um cliente de livraria tiver comprado livros de mistério no passado, ao visitar a página inicial da loja, ele provavelmente verá novos livros de suspense apresentados de forma destacada, enquanto um cliente mais voltado para culinária poderia ser recebido com novos livros de receitas. Como o site acompanha quem gosta de que, é algo que veremos mais adiante. Porém, resumindo, a resposta envolve cookies (até mesmo para visitantes que não gostam de culinária).

Na figura, o navegador entra em contato com três servidores para buscar as duas páginas, cs.washington.edu, youtube.com e google-analytics.com. O conteúdo desses

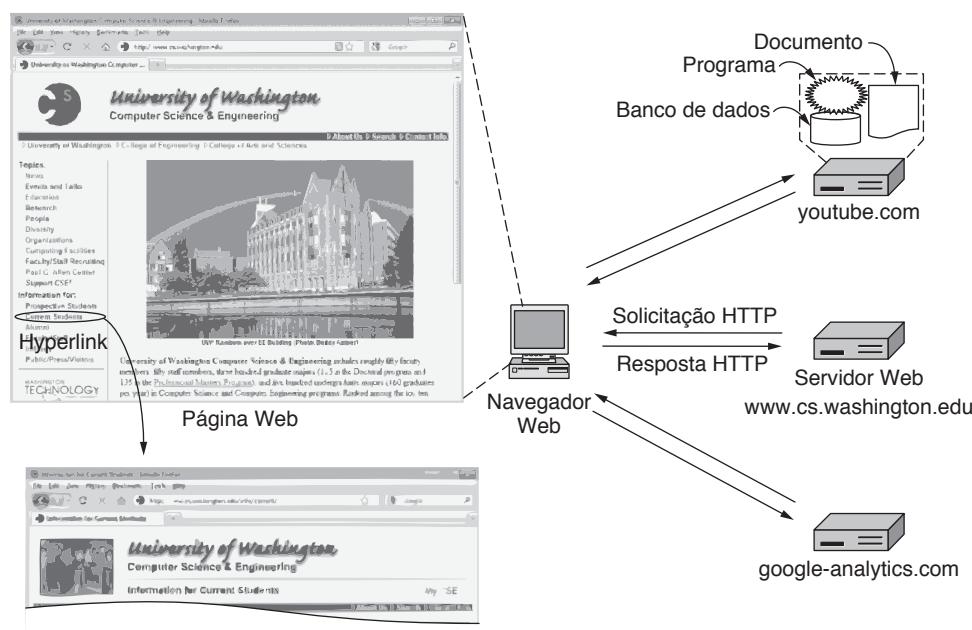



Figura 7.7 | Arquitetura da Web.

diferentes servidores é integrado para exibição pelo navegador. A exibição acarreta uma série de processamentos, que dependem do tipo de conteúdo. Além de apresentar texto e gráficos, um processamento pode envolver a exibição de um vídeo ou a execução de um script que apresenta sua própria interface com o usuário como parte da página. Nesse caso, o servidor cs.washington.edu fornece a página principal, o servidor youtube.com fornece um vídeo inserido e o servidor google-analytics.com não fornece nada que o usuário possa ver, mas acompanha os visitantes do site. Falaremos mais sobre trackers mais adiante.

O lado cliente

Agora vamos examinar o lado do navegador Web da Figura 7.7 com mais detalhes. Basicamente, um navegador é um programa que pode exibir uma página Web e capturar cliques do mouse em itens na página exibida. Quando um item é selecionado, o navegador segue o hiperlink e busca a página selecionada.

Quando a Web foi criada inicialmente, logo ficou aparente que ter uma página apontando para outra página Web exigia mecanismos para nomear e localizar páginas. Em particular, três perguntas tinham de ser respondidas antes que uma página selecionada pudesse ser exibida:

1. Como a página é chamada?
2. Onde a página está localizada?
3. Como a página pode ser acessada?

Se cada página recebesse de alguma forma um nome exclusivo, não haveria ambiguidade na identificação. Apesar disso, o problema não seria solucionado. Considere um paralelo entre as pessoas e as páginas. No Brasil, quase todos têm um número de CPF, que é um identificador exclusivo, de modo que duas pessoas não deveriam ter o mesmo número. Apesar disso, se você tiver apenas o número do CPF, não haverá como descobrir o endereço do seu proprietário, e certamente nenhuma forma de saber se você deve escrever para a pessoa em português, inglês, espanhol ou chinês. A Web tem basicamente os mesmos problemas.

A solução escolhida identifica as páginas de um modo que resolve os três problemas de uma só vez. Cada página recebe um **URL (Uniform Resource Locator)**, que efetivamente serve como o nome mundial da página. Os URLs têm três partes: o protocolo (também conhecido como o **esquema**), o nome DNS da máquina em que a página está localizada e o caminho que identifica exclusivamente a página específica (um arquivo para ler ou um programa para rodar na máquina). No caso geral, o caminho tem um nome hierárquico que modela uma estrutura de diretório de arquivo. Porém, a interpretação do caminho fica a critério do servidor; ele pode ou não refletir a estrutura de diretório real.

Como um exemplo, o URL da página mostrada na Figura 7.7 é

http://www.cs.washington.edu/index.html

Esse URL consiste em três partes: o protocolo (http), o nome DNS do host (www.cs.washington.edu) e o nome do caminho (index.html).

Quando um usuário clica em um hiperlink, o navegador executa uma série de etapas com o objetivo de buscar a página indicada. Vamos acompanhar as etapas que ocorrem quando esse link é selecionado.

1. O navegador determina o URL (verificando o que foi selecionado).
2. O navegador pergunta ao DNS qual é o endereço IP do servidor www.cs.washington.edu.
3. O DNS responde com 128.208.3.88.
4. O navegador estabelece uma conexão TCP com a porta 80 em 128.208.3.88, a porta conhecida para o protocolo HTTP.
5. Ele envia um comando HTTP solicitando a página /index.html.
6. O servidor www.cs.washington.edu envia a página como uma resposta HTTP, por exemplo, fornecendo o arquivo /index.html.
7. Se a página incluir URLs que são necessários para a exibição, o navegador busca os outros URLs usando o mesmo processo. Nesse caso, os URLs incluem várias imagens inseridas também capturadas de www.cs.washington.edu, um vídeo inserido do youtube.com e um script de google-analytics.com.
8. O navegador exibe a página /index.html conforme aparece na Figura 7.7.
9. As conexões TCP são fechadas se não houver outras solicitações para os mesmos servidores por um curto período.

Muitos navegadores exibem uma linha de status no rodapé da tela que indica qual etapa está sendo executada no momento. Dessa forma, quando o desempenho é fraco, o usuário pode verificar se a causa é falta de resposta do DNS, falta de resposta do servidor ou simplesmente congestionamento da rede durante a transmissão da página.

O projeto do URL tem a ponta aberta, no sentido de que é fácil fazer com que os navegadores usem vários protocolos para chegar a diferentes tipos de recursos. De fato, os URLs para diversos outros protocolos foram definidos. Formas ligeiramente simplificadas dos mais comuns são listadas na Tabela 7.9.

Vamos examinar a lista rapidamente. O protocolo *http* é a linguagem nativa da Web, aquela falada pelos servidores Web. **HTTP** significa **HyperText Transfer Protocol**. Vamos examiná-la com mais detalhes nesta seção.

O protocolo *ftp* é usado para acessar arquivos por FTP, o protocolo de transferência de arquivo da Internet. O FTP é anterior à Web, e foi usado por mais de três décadas. A Web facilita a obtenção de arquivos colocados em diversos

Nome	Usado para	Exemplo
http	Hipertexto (HTML)	http://www.ee.uwa.edu/~rob/
https	Hipertexto com segurança	https://www.bank.com/accounts/
ftp	FTP	ftp://ftp.cs.vu.nl/pub/minix/README
file	Arquivo local	file:///usr/suzana/prog.c
mailto	Envio de e-mail	mailto:JoaoSilva@acm.org
rtsp	Streaming de mídia	rtsp://youtube.com/montypython.mpg
sip	Chamadas de multimídia	sip:eve@adversary.com
about	Informação do navegador	about:plugins

Tabela 7.9 | Alguns esquemas de URL comuns.

servidores FTP no mundo inteiro, oferecendo uma interface simples e clicável em vez de uma interface por linha de comando. Esse acesso melhorado à informação é um motivo para o crescimento espetacular da Web.

É possível acessar um arquivo local como uma página Web usando o protocolo *file* ou, de forma mais simples, apenas nomeando-o. Essa técnica não exige um servidor. Naturalmente, ela só funciona para arquivos locais, e não remotos.

O protocolo *mailto* não tem realmente o glamour da busca de páginas Web, mas também é útil. Ele permite que os usuários enviem e-mail de um navegador web. A maioria dos navegadores responderá quando um enlace *mailto* for seguido pela partida do agente de correio do usuário para redigir uma mensagem com o campo do endereço já preenchido.

Os protocolos *rtsp* e *sip* são para estabelecimento de sessões de streaming de mídia e chamadas de áudio e vídeo.

Finalmente, o protocolo *about* é uma convenção que oferece informações sobre o navegador. Por exemplo, seguir o link *about:plugins* fará com que a maioria dos navegadores mostre uma página listando os tipos MIME que eles tratam como extensões do navegador chamadas plug-ins.

Resumindo, os URLs foram elaborados não apenas para permitir que os usuários naveguem pela Web, mas para executar protocolos mais antigos, como FTP e e-mail, bem como protocolos mais novos para áudio e vídeo, além de oferecer acesso conveniente a arquivos locais e informações do navegador. Essa técnica torna desnecessários todos os programas de interface com o usuário especializados para esses outros serviços, integrando quase todo o acesso à Internet em um único programa: o navegador Web. Se não fosse pelo fato de essa ideia ter sido imaginada por um físico britânico trabalhando em um laboratório de pesquisa na Suíça, ela poderia facilmente passar como um plano elaborado pelo departamento de propaganda de alguma empresa de software.

Apesar de todas essas propriedades interessantes, o uso crescente da Web se transformou em uma fraqueza inerente ao esquema do URL. Um URL aponta para um host específico, mas às vezes é útil referenciar uma página sem informar simultaneamente onde ela se encontra. Por exemplo, para páginas muito referenciadas, é desejável ter várias cópias distantes, para reduzir o tráfego da rede. Não há como dizer: 'Eu quero a página *xyz*, mas não importa onde você a busque'.

Para resolver esse tipo de problema, os URLs foram generalizados em **URIs** (**Uniform Resource Identifiers**). Alguns URIs informam como localizar um recurso. Estes são os URLs. Outros URIs dizem o nome de um recurso, mas não onde encontrá-lo. Esses URIs são chamados **URNs (Uniform Resource Names)**. As regras para a escrita de URIs são dadas na RFC 3986, embora os diferentes esquemas de URI em uso sejam acompanhados pela IANA. Existem muitos tipos de URIs além dos esquemas listados na Tabela 7.9, mas esses esquemas dominam a Web conforme é usada hoje.

Tipos MIME

Para permitir a exibição de uma nova página (ou qualquer página), o navegador precisa entender seu formato. Para permitir que todos os navegadores entendam todas as páginas Web, elas são escritas em uma linguagem padronizada chamada HTML. Essa é a língua internacional da Web (por enquanto). Vamos descrevê-la em detalhes mais adiante neste capítulo.

Embora um navegador seja basicamente um interpretador de HTML, a maioria dos navegadores tem diversos botões e recursos para facilitar a navegação na Web. A maioria tem um botão para voltar à página anterior, um botão para ir até a página seguinte (que só opera depois que o usuário volta dessa página) e um botão para ir direto à página inicial do próprio usuário. A maioria dos navegadores tem um botão ou um item de menu para definir um marcador (bookmark) em determinada página, e outro botão para exibir a lista de marcadores, possibilitando visitar outra vez qualquer página com apenas alguns cliques do mouse.

Como mostra o exemplo, páginas HTML podem conter elementos de conteúdo rico e não apenas texto e hipertex-

to. Para generalizar ainda mais, nem todas as páginas precisam conter HTML. Uma página pode consistir em um vídeo no formato MPEG, um documento em formato PDF, uma fotografia em formato JPEG, uma música em formato MP3 ou qualquer um de centenas de outros tipos de arquivos. Como as páginas HTML padrão podem se ligar a qualquer um destes, o navegador tem um problema quando alcança uma página a qual ele não sabe interpretar.

Em vez de tornar os navegadores cada vez maiores construindo interpretadores para uma coleção de tipos de arquivos que cresce com rapidez, a maioria dos navegadores opta por uma solução mais geral. Quando um servidor retorna uma página, ele também retorna algumas informações adicionais sobre a página. Essas informações incluem o tipo MIME da página (consulte a Tabela 7.6). Páginas do tipo *text/html* são exibidas diretamente, como também as páginas criadas em alguns outros tipos internos. Se o tipo MIME não for um dos tipos internos, o navegador consulta sua tabela de tipos MIME para saber como exibir a página. Essa tabela associa um tipo MIME a um visualizador.

Há duas possibilidades: plug-ins e aplicações auxiliares. Um plug-in é um módulo de código que o navegador busca em um diretório especial no disco e instala como uma extensão do próprio navegador, como ilustra a Figura 7.8(a). Alguns exemplos comuns são plug-ins para PDF, Flash e Quicktime, para exibir documentos e tocar áudio e vídeo. Como os plug-ins são executados dentro do navegador, eles têm acesso à página atual e podem modificar sua aparência.

Cada navegador tem um conjunto de procedimentos que todos os plug-ins devem implementar para que o navegador possa chamar o plug-in. Por exemplo, em geral existe um procedimento que o código básico do navegador chama para suprir o plug-in com dados para exibição. Esse conjunto de procedimentos é a interface do plug-in e é específico para o navegador.

Além disso, o navegador torna disponível um conjunto de seus próprios procedimentos para o plug-in, a fim de fornecer serviços aos plug-ins. Os procedimentos típicos na interface do navegador servem para alocar e liberar memória, exibir uma mensagem em sua linha de status e consultá-lo sobre parâmetros.

Antes de utilizarmos um plug-in, ele deve ser instalado. O procedimento habitual de instalação é fazer o usuário ir até o website do plug-in e baixar um arquivo de instalação. A execução do arquivo de instalação desempacota o plug-in e faz as chamadas apropriadas para registrar o tipo MIME e associá-lo a esse tipo. Os navegadores normalmente já vêm carregados com os plug-ins mais populares.

A outra maneira de estender um navegador é usar uma **aplicação auxiliar**, um programa completo que é executado como um processo separado. Essa aplicação é ilustrada na Figura 7.8(b). Tendo em vista que o auxiliar é um programa separado, ele está próximo do navegador. Em geral, ele simplesmente aceita o nome de um arquivo de rascunho em que o arquivo de conteúdo é armazenado, abre o arquivo e exibe o conteúdo. De modo geral, os auxiliares são grandes programas que existem de forma independente do navegador, como o Microsoft Word ou o PowerPoint.

Muitas aplicações auxiliares utilizam o tipo MIME *application*. Um número considerável de subtipos foi definido; por exemplo, application/vnd.ms-powerpoint para arquivos do PowerPoint. A extensão *vnd* indica formatos específicos do vendedor. Desse modo, um URL pode apontar diretamente para um arquivo do PowerPoint e, quando o usuário clicar sobre ele, o PowerPoint será inicializado automaticamente e receberá o conteúdo a ser exibido. As aplicações auxiliares não estão restritas a usar o tipo MIME *application*. O Adobe Photoshop usa *image/x-photoshop*, por exemplo.

Como consequência, os navegadores podem ser configurados para lidar com um número praticamente ilimitado de tipos de documentos, sem mudanças no navegador. Os servidores da Web modernos geralmente são configurados com centenas de combinações de tipo/subtipo, e novas combinações são acrescentadas toda vez que um novo programa é instalado.

Uma fonte de conflitos é que vários plug-ins e aplicações auxiliares estão disponíveis para alguns subtipos, como *video/mpeg*. O que acontece é que a última atualização a ser registrada modifica a associação existente com o tipo MIME, capturando o tipo para si. Como consequência, a instalação de um novo programa pode mudar o modo como um navegador trata os tipos existentes.

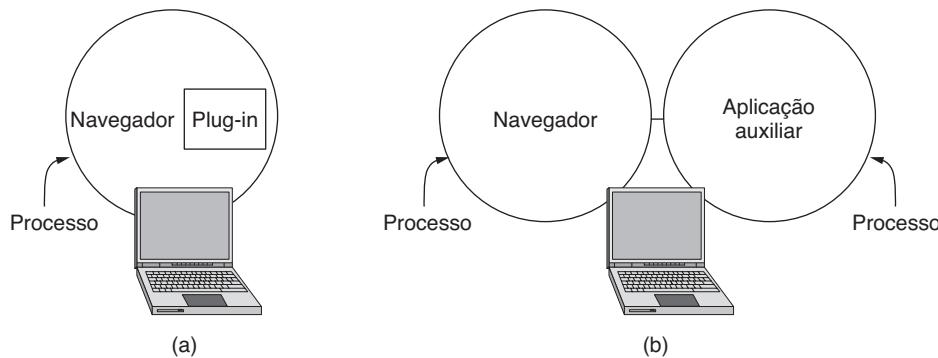



Figura 7.8 | (a) Um plug-in do navegador. (b) Uma aplicação auxiliar.

Os navegadores também podem abrir arquivos locais, em vez de buscá-los em servidores da Web remotos. Porém, o navegador precisa de alguma forma de determinar o tipo MIME do arquivo. O método-padrão é que o sistema operacional associe uma extensão do arquivo a um tipo MIME. Em uma configuração típica, abrir *foo.pdf* o abrirá no navegador usando um plug-in *application/pdf,* e abrir *bar.doc* o abrirá no Word como o auxiliar *application/msword*.

Nesse caso, também podem surgir conflitos, pois muitos programas estão dispostos — na verdade, ávidos — a tratar, digamos, arquivos mpg. Durante a instalação, programas destinados a profissionais com frequência exibem caixas de seleção com os tipos MIME e as extensões que eles estão preparados para manipular, a fim de permitir ao usuário selecionar as opções apropriadas e, assim, não sobrescrever associações existentes por engano. Os programas destinados ao mercado de consumo pressupõem que o usuário não tenha uma indicação de qual seja o tipo MIME, e simplesmente captam tudo que podem sem levar em consideração o que foi feito pelos programas já instalados.

A habilidade de estender o navegador com um grande número de novos tipos é conveniente, mas também pode causar problemas. Quando o PC busca um arquivo com a extensão *exe*, ele percebe que esse arquivo é um programa executável e, portanto, não tem auxiliar. A ação óbvia é executar o programa. Porém, isso poderia ser um enorme furo de segurança. Tudo que um site nocivo precisa fazer é produzir uma página Web com imagens de, por exemplo, astros do cinema ou heróis do esporte, todos vinculados a um vírus. Um único clique em uma imagem provoca a busca de um programa executável desconhecido e potencialmente hostil, seguida por sua execução na máquina do usuário. Para evitar convidados indesejáveis como esse, o Firefox e outros navegadores podem ser configurados para ser seletivos ao executar programas desconhecidos de forma automática, mas nem todos os usuários entendem quais escolhas são seguras em vez de convenientes.

O LADO SERVIDOR

Já estudamos o lado cliente. Agora, vamos examinar o lado servidor. Como vimos antes, quando o usuário digita um URL ou clica em uma linha de hipertexto, o navegador analisa o URL e interpreta a parte entre *http://* e a barra seguinte como um nome DNS a ser pesquisado. Munido do endereço IP do servidor, o navegador estabelece uma conexão TCP para a porta 80 desse servidor. Em seguida, ele envia um comando contendo o restante do URL, que é o caminho até a página nesse servidor. O servidor então retorna o arquivo para ser exibido pelo navegador.

Em linhas gerais, um servidor da Web é semelhante ao servidor do Quadro 6.1. Esse servidor recebe o nome de um arquivo para pesquisar e retornar pela rede. Em ambos os casos, as etapas que o servidor executa em seu loop principal são:

1. aceitar uma conexão TCP de um cliente (um navegador);
2. obter o caminho até a página, que é o nome do arquivo solicitado;
3. obter o arquivo (do disco);
4. enviar o conteúdo do arquivo ao cliente;
5. encerrar a conexão TCP.

Os servidores Web modernos têm outras características, mas, basicamente, é isso que um servidor da Web faz para o caso simples de conteúdo que está contido em um arquivo. Para o conteúdo dinâmico, a terceira etapa pode ser substituída pelo exemplo de um programa (determinado pelo caminho) que retorna o conteúdo.

No entanto, os servidores Web são implementados com um projeto diferente, para atender a mais solicitações por segundo. Um problema com o projeto simples é que o acesso aos arquivos normalmente é o gargalo. As leituras em disco são muito lentas em comparação com a execução do programa, e os mesmos arquivos podem ser lidos repetidamente do disco usando chamadas ao sistema operacional. Outro problema é que apenas uma solicitação é processada de cada vez. O arquivo pode ser grande, e outras solicitações serão bloqueadas enquanto ele é transferido.

Uma melhoria óbvia (usada por todos os servidores Web) é manter em cache na memória os n arquivos mais recentemente usados ou certo número de gigabytes de conteúdo. Antes de ir ao disco para obter um arquivo, o servidor verifica o cache. Se o arquivo estiver lá, ele poderá ser atendido diretamente da memória, eliminando assim o acesso ao disco. Embora o armazenamento efetivo em cache exija um grande volume de memória principal e algum tempo de processamento extra para verificá-lo e administrar seu conteúdo, a economia de tempo quase sempre compensa o overhead e as despesas.

Para enfrentar o problema de atender a uma única solicitação por vez, uma estratégia é tornar o servidor **multithreaded**. Em um projeto, o servidor consiste em um módulo de front end que aceita todas as solicitações recebidas e k módulos de processamento, como mostra a Figura 7.9. Os $k + 1$ threads pertencem todos ao mesmo processo, de forma que todos os módulos de processamento têm acesso ao cache dentro do espaço de endereços do processo. Ao chegar uma solicitação, o front end a recebe e cria um registro curto que a descreve. Em seguida, ele entrega o registro a um dos módulos de processamento.

O módulo de processamento primeiro verifica o cache para ver se o arquivo necessário está lá. Se estiver, ele atualiza o registro para incluir um ponteiro para o arquivo no registro. Se não estiver lá, o módulo de processamento inicia uma operação de disco para armazená-lo no cache (possivelmente descartando algum outro arquivo em cache para criar espaço). Quando o arquivo vem do disco, ele é colocado no cache e também é enviado de volta ao cliente.

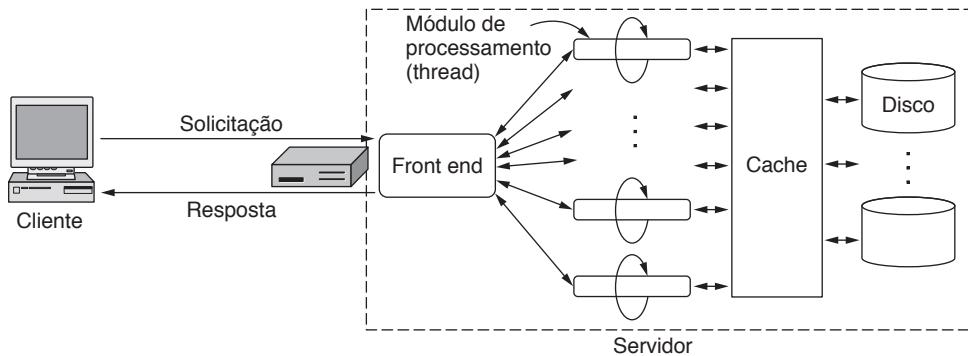



Figura 7.9 | Um servidor Web multithreaded com um front end e módulos de processamento.

A vantagem desse esquema é que, enquanto um ou mais módulos de processamento estão bloqueados esperando que uma operação de disco se complete (e, portanto, não consumindo tempo de CPU), outros módulos podem estar trabalhando ativamente em outras solicitações. Com *k* módulos de processamento, a vazão pode ser de até *k* vezes maior do que com um servidor de único thread. É claro que, quando o disco ou a rede é o fator limitador, é necessário ter vários discos ou uma rede mais rápida para obter qualquer melhoria real em relação ao modelo de único thread.

Os modernos servidores da Web fazem mais que apenas aceitar nomes de arquivos e retornar arquivos. De fato, o processamento real de cada solicitação pode ficar bastante complicado. Por essa razão, em muitos servidores, cada módulo de processamento executa uma série de etapas. O front end repassa cada solicitação recebida ao primeiro módulo disponível, que então a executa usando algum subconjunto das etapas a seguir, dependendo de quais delas sejam necessárias para essa solicitação específica. Essas etapas ocorrem depois de a conexão TCP e qualquer mecanismo de transporte seguro (como SSL/TSL, que serão descritos no Capítulo 8) ter sido estabelecidos.

1. Resolver o nome da página Web solicitada.
2. Executar o controle de acesso no cliente.
3. Verificar cache.
4. Buscar a página solicitada no disco ou executar um programa para montá-la.
5. Determinar o restante da resposta (por exemplo, o tipo MIME a enviar).
6. Retornar a resposta ao cliente.
7. Criar uma entrada no log do servidor.

A etapa 1 é necessária, porque a solicitação recebida talvez não possa conter o nome real do arquivo ou programa como uma string literal. Ela pode conter atalhos inseridos que precisam ser traduzidos. Como um exemplo simples, o URL http://www.cs.vu.nl/ tem um nome de arquivo vazio. Ele tem de ser expandido para formar algum nome de arquivo-padrão, que normalmente é *index.html*. Outra regra comum é mapear *~user/* para o diretório Web do *user*. Essas regras podem ser usadas juntas. Assim, a página inicial de um dos autores (AST) pode ser alcançada em

http://www.cs.vu.nl/~ast/

embora o nome de arquivo real seja *index.html* em um certo diretório-padrão.

Além disso, os navegadores modernos podem especificar informações de configuração como o software navegador e o idioma-padrão do usuário (por exemplo, português ou inglês). Isso torna possível para o servidor selecionar uma página Web com pequenas imagens para um dispositivo móvel e no idioma preferido, se estiver disponível. Em geral, a expansão de nome não é tão trivial quanto pode parecer a princípio, devido a uma série de convenções sobre como mapear caminhos para o diretório de arquivo e programas.

A etapa 2 verifica se quaisquer restrições de aceso associadas à página são cumpridas. Nem todas as páginas estão disponíveis ao público em geral. Determinar se um cliente pode buscar uma página pode depender da identidade do cliente (por exemplo, conforme dada por nomes de usuário e senhas) ou do local do cliente no espaço do DNS ou IP. Por exemplo, uma página pode ser restrita a usuários dentro de uma empresa. Como isso é feito depende do projeto do servidor. Para o servidor popular Apache, por exemplo, a convenção é colocar um arquivo chamado *.htaccess*, que lista as restrições de acesso no diretório onde a página restrita está localizada.

As etapas 3 e 4 envolvem capturar a página. Se ela pode ser capturada do cache, isso depende das regras de processamento. Por exemplo, páginas que são criadas executando programas nem sempre podem ser mantidas em cache, pois poderiam produzir um resultado diferente toda vez que fossem executadas. Até mesmo arquivos devem ser verificados ocasionalmente para saber se seu conteúdo mudou, de modo que o conteúdo antigo possa ser removido do cache. Se a página exigir um programa para ser executada, também há a questão de configurar os parâmetros ou a entrada do programa. Esses dados vêm do caminho ou de outras partes da solicitação.

A etapa 5 trata de determinar outras partes da resposta que acompanham o conteúdo da página. O tipo MIME é um exemplo. Ele pode vir de uma extensão de arquivo, das primeiras palavras do arquivo ou saída do programa, de um arquivo de configuração e possivelmente de outras fontes.

A etapa 6 é o retorno da página pela rede. Para aumentar o desempenho, uma única conexão TCP pode ser usada por um cliente e servidor para várias buscas de página. Essa reutilização significa que alguma lógica é necessária para mapear uma solicitação em uma conexão compartilhada e retornar cada resposta de modo que seja associada à solicitação correta.

A etapa 7 faz uma entrada no log do sistema para fins administrativos, com a manutenção de outras estatísticas importantes. Esses logs podem ser analisados mais tarde para gerar informações valiosas sobre o comportamento do usuário, por exemplo, a ordem em que as pessoas acessam as páginas.

Cookies

Navegar pela web conforme descrevemos até aqui envolve uma série de buscas de página independentes. Não existe o conceito de sessão de login. O navegador envia uma solicitação a um servidor e recebe um arquivo de volta. Depois, o servidor se esquece de ter visto esse cliente em particular.

Esse modelo é perfeitamente adequado para a leitura de documentos disponíveis publicamente, e funcionou bem quando a Web foi criada. Porém, ele não é adequado para retornar páginas diferentes para diferentes usuários, dependendo do que eles já fizeram com o servidor. Esse comportamento é necessário para muitas interações em andamento com os sites Web. Por exemplo, alguns sites (como os de jornais) exigem que os clientes se registrem (e possivelmente paguem algum dinheiro) para usá-los. Isso levanta a questão de como os servidores podem distinguir entre solicitações de usuários que se registraram anteriormente e todos os outros. Um segundo exemplo é o do comércio eletrônico (e-commerce). Se um usuário passear por uma loja virtual, colocando itens em seu carrinho de compras de vez em quando, como o servidor mantém o conteúdo do carrinho? Um terceiro exemplo são os portais personalizados da Web, como o Yahoo!. Os usuários podem montar uma página inicial detalhada personalizada, apenas com as informações que eles desejam (por exemplo, suas ações e seus times favoritos), mas como o servidor pode exibir a página correta se ele não sabe quem é o usuário?

Em princípio, pode-se pensar que os servidores poderiam acompanhar os usuários observando seus endereços IP. Porém, essa ideia não funciona. Muitos usuários compartilham computadores, especialmente em casa, e os endereços IP identificam apenas o computador, e não o usuário. Pior ainda, muitas empresas utilizam NAT, de modo que os pacotes que saem contêm o mesmo endereço IP para todos os usuários. Ou seja, todos os computadores por trás do NAT parecem iguais para o servidor. E muitos ISPs atribuem endereços IP a clientes com DHCP. Os endereços IP mudam com o tempo, de modo que, para um servidor, você de repente poderia se parecer com seu vizinho. Por todos esses motivos, o servidor não pode usar os endereços IP para registrar os usuários.

Esse problema é solucionado com um mecanismo frequentemente criticado, chamado **cookies**. O nome é derivado de uma antiga gíria de programador em que um programa chama um procedimento e recebe algo de volta que ele pode precisar apresentar mais tarde para conseguir algum trabalho. Nesse sentido, um descritor de arquivo do UNIX ou um handle de objeto do Windows podem ser considerados cookies. Os cookies foram implementados inicialmente no navegador Netscape em 1994, e agora são especificados na RFC 2109.

Quando um cliente solicita uma página Web, o servidor pode fornecer informações adicionais na forma de um cookie junto com a página solicitada. O cookie é uma string nomeada, bem pequena (no máximo 4KB), que o servidor pode associar a um navegador. Essa associação não é a mesma coisa que um usuário, mas é muito mais próxima e mais útil do que um endereço IP. Os navegadores armazenam os cookies oferecidos por conjunto, normalmente em um diretório de cookies no disco do cliente, de modo que os cookies existem entre as chamadas do navegador, a menos que o usuário os tenha desabilitado. Os cookies são apenas strings, e não programas executáveis. Em princípio, um cookie poderia conter um vírus, mas, como os cookies são tratados como dados, não existe um modo oficial para um vírus realmente ser executado e causar danos. Porém, sempre é possível que algum hacker explore um bug do navegador para causar uma ativação.

Um cookie pode conter até cinco campos, como mostra a Tabela 7.10. O *Domínio* informa de onde o cookie veio. Os navegadores deverão verificar se os servidores não estão mentindo sobre seu domínio. Cada domínio só poderá armazenar até 20 cookies por cliente. O *Caminho* é o percurso na estrutura de diretório do servidor que identifica quais partes da árvore de arquivos do servidor podem usar o cookie. Normalmente ele é /, que significa a árvore inteira.

O campo *Conteúdo* tem a forma *nome* = *valor*. Tanto *nome* quanto *valor* podem ser qualquer coisa que o servidor desejar. Esse campo é o local onde o conteúdo do cookie é armazenado.

O campo *Expira* especifica quando o cookie expira. Se esse campo não existir, o navegador descarta o cookie quando sair. Esse cookie é chamado **cookie não persistente**. Se hora e data forem indicados, o cookie é considerado um **cookie persistente**, e é mantido até que expire. As horas de expiração são dadas na hora GMT. Para remover um cookie do disco rígido de um cliente, um servidor apenas o envia novamente, mas com uma data de expiração anterior.

Domínio	Caminho	Conteúdo	Expira	Seguro
toms-casino.com	/	CustomerI = D297793521	15-10-10 17:00	Sim
jills-store.com	/	Cart = 1-00501;1-07031;2-13721	11-1-11 14:22	Não
aportal.com	/	Prefs = Stk:CSCO + ORCL;Stp:Jets	31-12-20 23:59	Não
sneaky.com	/	UserID = 4627239101	31-12-19 23:59	Não

Tabela 7.10 | Alguns exemplos de cookies.

Finalmente, o campo *Seguro* pode ser definido para indicar que o navegador só pode retornar o cookie para um servidor usando um transporte seguro, a saber, SSL/TSL (que descreveremos no Capítulo 8). Esse recurso é usado para comércio eletrônico, operações bancárias e outras aplicações seguras.

Agora já vimos como os cookies são adquiridos, mas como eles são usados? Antes que um navegador envie uma solicitação para uma página em algum site, ele verifica seu diretório de cookies para ver se foram incluídos cookies pelo domínio para onde a solicitação está indo. Nesse caso, todos os cookies colocados por esse domínio, e somente esse domínio, estão incluídos na mensagem de solicitação. Quando o servidor os recebe, ele pode interpretá-los como desejar.

Vamos examinar alguns usos possíveis para os cookies. Na Tabela 7.10, o primeiro cookie foi definido por toms-casino.com e é usado para identificar o cliente. Quando o cliente retornar na próxima semana para gastar mais dinheiro, o navegador envia o cookie para que o servidor saiba quem ele é. De posse do ID do cliente, o servidor pode pesquisar o registro em um banco de dados e usar essa informação para montar uma página Web apropriada para exibição. Dependendo dos hábitos de jogo conhecidos do cliente, essa página poderia consistir em uma rodada de pôquer, uma listagem das corridas de cavalo do dia ou uma máquina caça-níqueis.

O segundo veio de jills-store.com. Nesse cenário, o cliente está passeando pela loja, procurando coisas boas para comprar. Quando ele encontra uma oferta e clica nela, o servidor a acrescenta em seu carrinho de compras (mantido no servidor) e também monta um cookie contendo o código de produto do item e o envia de volta ao cliente. Quando o cliente continua a passear pela loja clicando em novas páginas, o cookie é retornado ao servidor em cada nova solicitação de página. À medida que mais compras são acumuladas, o servidor as inclui no cookie. Finalmente, quando o cliente clica em PROSSEGUIR PARA O CAIXA, o cookie, agora contendo a lista completa de compras, é enviado com a solicitação. Desse modo, o servidor sabe exatamente o que o cliente quer comprar.

O terceiro cookie é para um portal Web. Quando o cliente clica em um link para o portal, o navegador envia o cookie. Isso diz ao portal para montar uma página contendo os preços de ações da Cisco e da Oracle, e os resultados de futebol do Palmeiras. Como um cookie pode ter até 4 KB, há muito espaço para preferências mais detalhadas sobre manchetes de jornais, previsão de tempo local, ofertas especiais etc.

Um uso controvertido dos cookies tem a finalidade de reunir informações sobre os hábitos de navegação dos usuários. Os operadores de site entendem como os usuários navegam e os anunciantes acumulam perfis dos anúncios ou sites que determinado usuário visitou. A controvérsia é que os usuários normalmente não sabem que sua atividade está sendo monitorada, até mesmo com perfis detalhados e por sites aparentemente não relacionados. Apesar disso, o **rastreamento na Web** é uma atividade muito grande. DoubleClick, que oferece e rastreia anúncios, está avaliada entre os cem maiores sites do mundo pela empresa de monitoramento da Web Alexa. Google Analytics, que rastreia o uso dos sites para os operadores, é usado por mais da metade dos 100 mil sites mais utilizados na Web.

É muito fácil para um servidor rastrear a atividade do usuário com cookies. Suponha que um servidor queira rastrear quantos visitantes diferentes ele teve e quantas páginas cada visitante viu antes de sair do site. Quando chega a primeira solicitação, não haverá um cookie correspondente, de modo que o servidor envia um cookie contendo *Contador = 1*. Outras visões de página nesse site enviarão o cookie de volta ao servidor. A cada vez, o contador é incrementado e enviado de volta ao cliente. Acompanhando os contadores, o servidor pode ver quantas pessoas saíram depois de ver a primeira página, quantas viram duas páginas e assim por diante.

O rastreamento do comportamento de navegação dos usuários entre os sites é apenas um pouco mais complicado. Ele funciona da maneira ilustrada a seguir. Uma agência de publicidade, digamos, a Sneaky Ads, entra em contato com sites importantes e insere banners anunciando os produtos de seus clientes corporativos nas páginas desses sites, pagando uma taxa aos proprietários. Em vez de fornecer a propaganda ao site como um arquivo GIF para ser colocado em cada página, a agência lhes entrega um URL que será incluído em cada página. Cada URL entregue contém um número exclusivo no caminho, como:

http://www.sneaky.com/382674902342.gif

Quando um usuário visita pela primeira vez uma página *P* que contém esse anúncio, o navegador busca o arquivo HTML. Depois, examina o arquivo HTML e encontra o link para o arquivo de imagem em www.sneaky.com e, em seguida, solicita a imagem. Um arquivo GIF contendo um anúncio é retornado, juntamente com um cookie contendo uma ID de usuário exclusiva, 4627239101, na Tabela 7.10. A Sneaky registra o fato de que o usuário com essa ID visitou a página *P*. É fácil fazer isso, pois o arquivo solicitado (*382674902342.gif*) só é referenciado na página *P*. É claro que o anúncio real pode aparecer em milhares de páginas, mas cada vez com um nome de arquivo diferente. É provável que a Sneaky ganhe alguns centavos do fabricante do produto toda vez que transmite o anúncio.

Mais tarde, quando o usuário visitar outra página contendo qualquer anúncio da Sneaky, o navegador primeiro busca o arquivo HTML no servidor. Depois, ele encontra o link para, digamos, http://www.sneaky.com/193654919923.gif na página e solicita esse arquivo. Como ele já tem um cookie do domínio sneaky.com, o navegador inclui o cookie da Sneaky contendo a ID do usuário. Agora a Sneaky sabe qual foi a segunda página que o usuário visitou.

Depois de algum tempo, a Sneaky poderá elaborar um perfil completo dos hábitos de navegação do usuário, ainda que ele jamais tenha clicado em algum dos anúncios. É claro que a agência ainda não tem o nome do usuário (embora tenha seu endereço IP, o que pode ser suficiente para deduzir o nome a partir de outros bancos de dados). Porém, se o usuário fornecer seu nome a algum site que coopere com a Sneaky, haverá um perfil completo juntamente com um nome disponível para ser vendido a quem quiser comprá-lo. A venda dessas informações pode ser lucrativa o suficiente para a Sneaky colocar mais anúncios em mais sites e assim coletar mais informações.

E se a Sneaky quiser ser superfurtiva, o anúncio não precisa ser um banner clássico. Um 'anúncio' consistindo em um único pixel na cor de fundo (e, desse modo, invisível) tem exatamente o mesmo efeito de um banner: ele exige que o navegador vá buscar a imagem GIF de 1 × 1 pixel e envie todos os cookies originários do domínio do pixel.

Os cookies se tornaram um ponto focal para o debate sobre a privacidade on-line, devido ao comportamento de rastreamento que explicamos. A parte mais traiçoeira dessa atividade inteira é que muitos usuários estão completamente desavisados dessa coleta de informação e podem até pensar que estão seguros, pois não clicaram em nenhum anúncio. Por esse motivo, os cookies que rastreiam os usuários entre os sites são classificados por muitos como **spyware**. Dê uma olhada nos cookies que já foram armazenados pelo seu navegador. A maioria dos navegadores exibirá essa informação junto com as preferências de privacidade atuais. Você poderá ficar surpreso ao encontrar nomes, endereços de e-mail ou senhas, além de identificadores irreconhecíveis. Com sorte, você não achará números de cartão de crédito, mas o potencial de abuso é evidente.

Para manter alguma aparência de privacidade, alguns usuários configuram seus navegadores para rejeitar todos os cookies. Porém, isso pode trazer problemas com sites legítimos, que precisam usar cookies. Como alternativa, a maioria dos navegadores permite que os usuários bloqueiem **cookies de terceiros**. Um cookie de terceiros é aquele de um site diferente do da página principal que está sendo buscada — por exemplo, o cookie sneaky.com que é usado na interação com a página *P* em um site completamente diferente. Impedi-lo ajuda a evitar o rastreamento entre os sites. Extensões do navegador também podem ser instaladas para fornecer controle minucioso sobre o modo como os cookies são usados (ou, então, não são usados). À medida que o debate continua, muitas empresas estão desenvolvendo políticas de privacidade que limitam o modo como elas compartilharão as informações, para evitar abusos. É claro que as políticas são simplesmente o modo como as empresas dizem que tratarão da informação. Por exemplo, 'Podemos usar a informação que você fornece na condução de nosso negócio' — que pode significar vender a informação.

7.3.2 Páginas Web estáticas

A base da Web é a transferência de páginas do servidor para o cliente. Em sua forma mais simples, elas são estáticas, isto é, são apenas arquivos que ficam armazenados em algum servidor da mesma maneira toda vez que são buscados e exibidos. Contudo, só porque são estáticos, não significa que as páginas estão inertes no navegador. Uma página contendo um vídeo pode ser uma página Web estática.

Como já dissemos, a língua internacional da Web, em que a maioria das páginas é escrita, é HTML. As páginas iniciais de professores normalmente são páginas Web HTML estáticas. As páginas iniciais de empresas normalmente são dinâmicas, montadas por uma empresa de projeto da Web. Nesta seção, daremos uma rápida olhada nas páginas HTML estáticas como um alicerce para o material seguinte. Os leitores acostumados com HTML poderão saltar para a seção seguinte, na qual descrevemos o conteúdo dinâmico e os Web services.

HTML – HyperText Markup Language

A linguagem de marcação de hipertexto, ou **HTML (HyperText Markup Language)**, foi introduzida com a Web. Ela permite que os usuários produzam páginas que incluem texto, gráficos, vídeo, ponteiros para outras páginas Web e muito mais. A HTML é uma linguagem de marcação, ou seja, uma linguagem para descrever como os documentos devem ser formatados. O termo 'marcação' vem da época em que os editores realmente marcavam os documentos para informar ao impressor — naquele tempo, uma pessoa — que fontes usar e assim por diante. Portanto, as linguagens de marcação contêm comandos explícitos de formatação. Por exemplo, em HTML, **significa início**

do modo negrito, e significa fim do modo negrito. LaTeX e TeX são outros exemplos de linguagens de marcação, bem conhecidos da maioria dos autores acadêmicos.

A principal vantagem de uma linguagem de marcação em comparação com uma sem marcação explícita é que ela separa o conteúdo do modo como ele deve ser apresentado. Escrever um navegador, então, é muito simples: o navegador simplesmente precisa entender os comandos de marcação e aplicá-los ao conteúdo. Embutir todos os comandos de marcação dentro de cada arquivo HTML e padronizá-los permite que qualquer navegador leia e reformate qualquer página Web. Isso é fundamental, pois uma página pode ter sido produzida em uma janela de 1600 × 1200 com 24 bits de cores em um computador de alto nível, mas talvez tenha que ser exibida em uma janela de 640 × 320 em um telefone móvel.

Embora certamente seja possível escrever documentos dessa forma com qualquer editor de textos comum — e muitas pessoas assim o fazem —, também é possível usar processadores de textos ou editores de HTML especiais, que realizam a maior parte do trabalho (porém, dão ao usuário menos controle direto sobre os detalhes do resultado final).

Uma página Web simples escrita em HTML e sua apresentação em um navegador podem ser vistas no Quadro 7.4. Uma página Web consiste em um cabeçalho e um corpo entre as **tags** (comandos de formatação) <html> e </html>, embora a maioria dos navegadores não reclame se essas tags não estiverem presentes. Como podemos ver no Quadro 7.4(a), o cabeçalho começa e termina com as tags <head> e </head>, respectivamente, enquanto o corpo é delimitado pelas tags <body> e </body>. As strings entre as tags são chamadas **diretivas**. A maioria das tags HTML, mas não todas, tem esse formato, ou seja, <algo> para marcar o início de alguma coisa e </algo> para marcar seu fim.

As tags não fazem distinção entre maiúsculas e minúsculas. Portanto, <head> e <HEAD> têm o mesmo significado, embora o uso de minúsculas seja melhor, por compatibilidade. O verdadeiro layout do documento HTML é irrelevante. Os analisadores de HTML ignoram espaços em branco e retornos de cursor, pois têm de reformatar o texto de modo a encaixá-lo na área de exibição atual. Consequentemente, o espaço em branco pode ser acrescentado à vontade para tornar os documentos HTML mais legíveis, algo de que a maioria dos documentos precisa muito. Como outra consequência, linhas em branco não podem ser usadas para separar parágrafos, pois simplesmente são ignoradas. Uma tag explícita precisa ser usada.

Algumas tags possuem parâmetros (nomeados), chamados **atributos**. Por exemplo, a tag <img> no Quadro 7.4 é usada para incluir uma imagem imediatamente seguida ao texto. Ela tem dois atributos, src e alt. O primeiro atributo indica o URL para a imagem. O padrão HTML não especifica quais formatos de imagem são permitidos. Na prática, todos os navegadores aceitam arquivos GIF e JPEG. Os navegadores são livres para admitir outros formatos, mas essa extensão é uma espada de dois gumes. Se um usuário estiver acostumado com um navegador que admite, digamos, arquivos TIFF, ele pode incluí-los em suas páginas e depois ficar surpreendido com outros navegadores que simplesmente ignoram toda a sua arte maravilhosa.

O segundo atributo fornece o texto alternativo que é usado se a imagem não puder ser exibida. Para cada tag, o padrão HTML oferece uma lista dos parâmetros permitidos, se houver, e o que significam. Como cada parâmetro tem um nome, a ordem em que eles aparecem não importa.

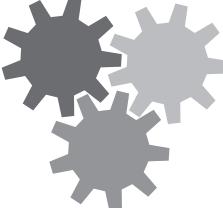
Tecnicamente, os documentos HTML são escritos com o conjunto de caracteres Latin-1 da ISO 8859-1, mas, para os usuários cujos teclados aceitam apenas ASCII, existem sequências alternativas para caracteres especiais, como o caractere è. A lista de caracteres especiais é informada no padrão. Todos eles começam com um símbolo & e terminam com um ponto-e-vírgula. Por exemplo,   produz um espaço, è produz è e é produz é. Como <,> e & possuem significados especiais, eles só podem ser expressos com suas sequências de escape, <, > e &, respectivamente.

O item principal no cabeçalho é o título, delimitado por <title> e </title>. Certos tipos de metainformação também podem estar presentes, embora isso não ocorra em nosso exemplo. O próprio título não é exibido na página. Os navegadores o utilizam para rotular a janela da página.

Vários cabeçalhos são usados no Quadro 7.4. Cada um é gerado por uma tag <h*n*>, onde *n* é um dígito no intervalo de 1 a 6. Assim, <h1> é o cabeçalho mais importante; <h6> é o menos importante. Fica a critério do navegador exibir esses cabeçalhos de modo apropriado na tela. Normalmente, os cabeçalhos com números mais baixos serão exibidos em uma fonte maior e mais espessa. O navegador também pode decidir usar cores diferentes para cada nível de cabeçalho. Normalmente, os cabeçalhos <h1> são grandes e em negrito, com pelo menos uma linha em branco acima e abaixo. Ao contrário, os cabeçalhos <h2> usam uma fonte menor, com menos espaço acima e abaixo.

As tags <b> e <i> são usadas para dar início aos modos negrito e itálico, respectivamente. A tag <hr> força uma quebra e linha e desenha um traço horizontal na tela.

A tag <p> inicia um parágrafo. O navegador pode exibir isso inserindo uma linha em branco e algum recuo, por exemplo. O interessante é que a tag </p>, que existe para marcar o final de um parágrafo, normalmente é omitida por programadores HTML menos exigentes.

A HTML oferece diversos mecanismos para criar listas, incluindo listas com recuos. As linhas não ordenadas, como aquelas no Quadro 7.4, são iniciadas com <ul>, com tags <li> usadas para marcar o início dos itens. Há também uma tag <ol> para iniciar uma lista ordenada. Os itens individuais nas listas não ordenadas geralmente aparecem com marcadores (●) à sua frente. Os itens nas listas ordenadas são numerados pelo navegador.

```
<html>
<head> <title> AMALGAMATED WIDGET, INC. </title> </head>
<body> <h1> Welcome to AWI’s Home Page </h1>
<img src = “http://www.widget.com/images/logo.gif = “ALT”AWI Logo”> <br>
We are so happy that you have chosen to visit <b> Amalgamated Widget’s</b>
home page. We hope <i> you </i> will find all the information you need here.
<p>Below we have links to information about our many fine products.
You can order electronically (by WWW), by telephone, or by email. </p>
<hr>
<h2> Product information </h2>
<ul>
    <li> <a href = “http://widget.com/products/big”> Big widgets </a> </li>
    <li> <a href = “http://widget.com/products/little”> Little widgets </a> </li>
</ul>
<h2> Contact information </h2>
<ul>
    <li> By telephone: 1-800-WIDGETS </li>
    <li> By email: info@amalgamated-widget.com </li>
</ul>
</body>
</html>
```

(a)

Welcome to AWI's Home Page

We are so happy that you have chosen to visit **Amalgamated Widget's** home page. We hope *you* will find all the information you need here.

Below we have links to information about our many fine products. You can order electronically (by WWW), by telephone, or by email.

Product Information

- Big widgets
- Little widgets

Contact information

- By telephone: 1-800-WIDGETS
- By email: info@amalgamated-widget.com

(b)

Quadro 7.4 | (a) A HTML para uma página Web exemplo. (b) A página formatada.

Por último, chegamos aos hiperlinks. Alguns exemplos deles podem ser vistos no Quadro 7.4 utilizando as tags <a> (âncora) e . A tag <a> tem vários parâmetros, sendo o mais importante *href* (o URL vinculado). O texto entre <a> e é apresentado na tela. Se for selecionado, o hiperlink levará a uma nova página. Também é permitido vincular a

outros elementos. Por exemplo, pode ser incluída uma imagem entre <a> e usando <img>; nesse caso, a imagem é exibida e um clique sobre ela também ativa o hiperlink.

Existem muitas outras tags HTML e atributos que não vimos neste exemplo simples. Por exemplo, a tag <a> pode usar o parâmetro *name* para inserir um hiperlink, permitindo que ele aponte para o meio de uma página. Isso é útil, por exemplo, para páginas Web que começam com uma lista clicável. Clicando em um item nessa lista, o usuário salta para a seção correspondente da mesma página. Um exemplo de uma tag diferente é <br>. Ela força o navegador a quebrar a linha e iniciar uma nova linha.

Provavelmente, a melhor maneira de entender as tags é examiná-las em ação. Para fazer isso, você pode escolher uma página e examinar a HTML no seu navegador, para ver como a página foi criada. A maioria dos navegadores possui um item de menu EXIBIR CÓDIGO FONTE (ou algo semelhante). A seleção desse item apresenta o código fonte HTML da página atual, em vez de sua saída formatada.

Já mostramos superficialmente as tags que existem desde o início da Web. Mas a HTML continua evoluindo. A Tabela 7.11 mostra alguns dos recursos que foram acrescentados às versões sucessivas da HTML. HTML 1.0 refere-se à versão usada com a introdução da Web. HTML versões 2.0, 3.0 e 4.0 apareceram em rápida sucessão no espaço de apenas alguns anos, à medida que a Web explodiu. Após a HTML 4.0, um período de quase dez anos se passou antes que o caminho para a padronização da próxima versão principal, HTML 5.0, se tornasse claro. Por ser uma atualização importante, que consolida as formas como os navegadores tratam de conteúdo rico, o esforço para a HTML 5.0 está em andamento e não deverá produzir um padrão antes de 2012, no mínimo. Independentemente dos padrões, os principais navegadores já admitem a funcionalidade da HTML 5.0.

A progressão pelas versões da HTML permitiu a inclusão de novos recursos que as pessoas queriam, mas que tinham de tratar de maneiras não padronizadas (por exemplo, plug-ins) até que se tornaram padrão. Por exemplo, HTML 1.0 e HTML 2.0 não tinham tabelas. Elas foram acrescentadas na HTML 3.0. Uma tabela HTML consiste em uma ou mais linhas, cada uma consistindo em uma ou mais células que podem conter uma grande variedade de material (por exemplo, texto, imagens, outras tabelas). Antes da HTML 3.0, os autores tinham que lançar mão de métodos alternativos, como a inclusão de uma imagem mostrando a tabela.

Item	HTML 1.0	HTML 2.0	HTML 3.0	HTML 4.0	HTML 5.0
Hiperlinks	×	×	×	×	×
Imagens	×	×	×	×	×
Listas	×	×	×	×	×
Mapas e imagens ativas		×	×	×	×
Formulários		×	×	×	×
Equações			×	×	×
Barras de ferramentas			×	×	×
Tabelas			×	×	×
Recursos de acessibilidade				×	×
Objetos inseridos				×	×
Folhas de estilo				×	×
Scripting				×	×
Vídeo e áudio					×
Gráficos de vetores em linha					×
Representação XML					×
Threads em segundo plano					×
Armazenamento pelo navegador					×
Tela de desenho					×

Tabela 7.11 | Algumas diferenças entre as versões da HTML.

Na HTML 4.0, mais recursos novos foram acrescentados. Entre eles estão os recursos de acessibilidade para usuários fisicamente incapacitados, incorporação de objetos (uma generalização da tag <img>, para que outros objetos também pudessem ser incorporados às páginas), suporte para linguagens de scripting (para permitir conteúdo dinâmico) e outros.

A HTML 5.0 inclui muitos recursos para lidar com a mídia rica que agora é usada como rotina na Web. Vídeo e áudio podem ser incluídos em páginas e reproduzidos pelo navegador sem exigir que o usuário instale plug-ins. Desenhos como gráficos de vetores podem ser montados no navegador, em vez do uso de formatos de imagem com mapas de bits (como JPEG e GIF). Também há mais suporte para executar scripts nos navegadores, como threads de computação em segundo plano e acesso ao armazenamento. Todos esses recursos ajudam a dar suporte às páginas Web que são mais aplicações tradicionais, com uma interface com o usuário, em vez de documentos. Essa é a tendência atual da Web.

Entrada e formulários

Há uma capacidade importante que ainda não discutimos: entrada. A HTML 1.0 era basicamente unidirecional. Os usuários poderiam buscar páginas dos provedores de informação, mas era difícil enviar informações de volta. Rapidamente ficou claro que havia uma necessidade de um tráfego bidirecional, para permitir que pedidos de produtos fossem feitos por páginas Web, cartões de registro fossem preenchidos on-line, termos de busca fossem inseridos e muito, muito mais.

Enviar a entrada do usuário para o servidor (via navegador) exige dois tipos de suporte. Primeiro, é preciso que o HTTP seja capaz de transportar dados nessa direção. Descrevemos como isso é feito em outra seção; é usado o método *POST*. O segundo requisito é poder apresentar os elementos da interface com o usuário que reúnem e empacotam a entrada. Os **formulários** foram incluídos com essa funcionalidade na HTML 2.0.

Formulários contêm caixas ou botões que permitem que os usuários preencham informações ou façam escolhas e depois enviem a informação de volta ao proprietário da página. Os formulários são escritos como outras partes da HTML, como vemos no exemplo do Quadro 7.5. Observe que os formulários ainda são conteúdo estático. Eles exibem o mesmo comportamento, não importa quem os utilize. O conteúdo dinâmico, que veremos mais adiante, oferece maneiras mais sofisticadas de colher entrada, enviando um programa cujo comportamento pode depender do ambiente do navegador.

Como todos os formulários, este está delimitado pelas tags <form> e </form>. Os atributos dessas tags dizem o que fazer com os dados que são informados, neste caso —, usar o método *POST* para enviar os dados ao URL especificado. O texto não delimitado em uma tag é apenas exibido. Todas as tags normais (por exemplo, <b>) são permitidas em um formulário, para que o autor da página controle a aparência do formulário na tela.

Três tipos de caixas de entrada são usados nesse formulário, cada um adotando a tag <input>. Ela se vale de diversos parâmetros para determinar o tamanho, a natureza e o uso da caixa exibida. Os formulários mais comuns são campos em branco para aceitar texto do usuário, caixas que podem ser marcadas e botões de *envio*, que fazem com que os dados sejam retornados ao servidor.

O primeiro tipo de caixa de entrada é uma caixa de *texto*, que vem após o texto 'Name'. A caixa tem 46 caracteres de largura e espera que o usuário digite um texto, que é então armazenado na variável *customer*.

A próxima linha, para o endereço do usuário, tem 40 caracteres de largura. Em seguida, vem uma linha que solicita os nomes da cidade, do estado e do país. A tag <p> não é usada entre esses campos; portanto, o navegador exibirá todos eles na mesma linha, se houver espaço (em parágrafos separados). Do ponto de vista do navegador, esse parágrafo contém apenas seis itens: três strings que se alternam com três caixas. A linha seguinte pede o número de cartão de crédito e a data de validade. A transmissão de números de cartão de crédito pela Internet só deve ser feita com mecanismos de segurança adequados. Discutiremos alguns deles no Capítulo 8.

Depois da data de validade, encontramos um novo recurso: os botões de rádio. Eles são utilizados quando é necessário optar entre duas ou mais alternativas. O modelo imaginário aqui é um rádio de carro com meia dúzia de botões para sintonizar as estações. Ao clicar sobre um botão, o usuário desativa todos os outros do mesmo grupo. A apresentação visual cabe ao navegador. O campo Widget size também utiliza dois botões de rádio. Os dois grupos se distinguem por seu parâmetro *name*, e não por um escopo estático que utilize algo como <radiobutton> ... </radiobutton>.

Os parâmetros *value* são usados para indicar que botão de rádio foi pressionado. Por exemplo, dependendo de qual das opções de cartão de crédito o usuário selecionou, a variável *cc* será definida como a string 'mastercard' ou 'visacard'.

Depois dos dois conjuntos de botões de rádio, chegamos à opção de transporte, representada por uma caixa de seleção do tipo *checkbox*. Ela pode estar selecionada ou não. Ao contrário dos botões de rádio, nos quais apenas uma opção deve ser escolhida, cada caixa de seleção do tipo *checkbox* pode ser ou não selecionada, de forma independente de todas as outras.

Finalmente, chegamos ao botão *submit*. A string *value* é o rótulo no botão, que é exibido ao usuário. Quando este clica no botão *submit*, o navegador empacota a informação coletada em uma única linha longa e a envia de volta ao servidor, para o URL fornecido como parte da tag <form>. Para o nosso exemplo de formulário, a linha poderia se parecer com o conteúdo do Quadro 7.6.

```
<html>
<head> <title> AWI CUSTOMER ORDERING FORM </title> </head>
<body>
<h1> Widget Order Form </h1>
<form ACTION = "http://widget.com/cgi-bin/order.cgi" method = POST>
<p> Name <input name = "customer" size = 46 </p>
<p> Street address <input name = "address" size = 40> </p>
<p> City <input name = "city" size = 20> State <input name = "state" size = 4>
Country <input name = "country" size = 10> </p>
<p> Credit card # <input name = "cardno" size = 10>
Expires <input name = "expires" size = 4>
M/C <input name = "cc" type = radio value = "mastercard">
VISA <input name = "cc" type = radio value = "visacard"> </p>
<p> Widget size Big <input name = "product" type = radio value = "expensive">
Little <input name = "product" type = radio value = "cheap">
Ship by express courier <input name = "express" type = checkbox> </p>
<p> <input type = submit value = "Submit order"> </p>
Thank you for ordering an AWI widget, the best widget money can buy!
</form>
</body>
</html>
```

(a)

Widget Order Form

Name

Street address

City State Country

Credit card # Expires M/C Visa

Widget size Big Little Ship by express courier

Submit order

Thank you for ordering an AWI widget, the best widget money can buy!

(b)

Quadro 7.5 | (a) Uma HTML para um formulário de pedido. (b) A página formatada.

```
customer = John + Doe&address = 100 + Main + St.&city = White + Plains&
state = NY&country = USA&cardno = 1234567890&expires = 6/14&cc = mastercard&
product = cheap&express = on
```

Quadro 7.6 | Uma resposta possível do navegador ao servidor, com informações preenchidas pelo usuário.

A string é enviada de volta ao servidor como uma linha. (Ela está dividida em três linhas aqui porque a página não é larga o suficiente.) Fica a critério do servidor entender o sentido dessa string, provavelmente passando informações a um programa, que a processará. Discutiremos como isso pode ser feito na próxima seção.

Também existem outros tipos de entrada que não aparecem nesse exemplo simples. Dois outros tipos são *password* e *textarea*. Uma caixa *password* é o mesmo que uma caixa *text* (o tipo-padrão que não precisa ser nomeado), exceto que os caracteres não aparecem enquanto são digitados. Uma caixa *textarea* também é o mesmo que uma caixa *text*, exceto que pode conter várias linhas.

Cabe observar que para listas muito longas, a partir das quais se deve escolher um item, as tags <select> e </select> são fornecidas para delimitar uma lista de alternativas. Essa lista normalmente é apresentada como um menu dropdown. A semântica é a mesma daquela usada por botões de rádio, a menos que lhe seja dado o parâmetro *multiple*; nesse caso, a semântica segue o padrão das caixas de seleção.

Finalmente, existem maneiras de indicar os valores default ou iniciais que o usuário pode mudar. Por exemplo, se uma caixa *text* receber um campo *value*, o conteúdo é exibido no formulário para o usuário editar ou apagar.

CSS – Cascading Style Sheets

O objetivo original da HTML foi especificar a *estrutura* do documento, e não sua *aparência*. Por exemplo,

```
<h1> Deborah's Photos </h1>
```

instrui o navegador a enfatizar o cabeçalho, mas não indica nada sobre a fonte, o tamanho ou a cor. Isso fica para o navegador, que conhece as propriedades da exibição (por exemplo, quantos pixels ele tem). Porém, muitos projetistas de página Web desejam ter controle absoluto sobre o modo como suas páginas aparecem, de forma que novas tags foram acrescentadas à HTML para controlar a aparência, como

```
<font face = "helvetica" size = "24" color = "red"> Deborah's Photos </font>
```

Além disso, foram incluídas maneiras de controlar o posicionamento da tela com precisão. O problema com essa técnica é que ela é tediosa e produz um código HTML muito cheio, que não é portável. Embora uma página possa ser apresentada perfeitamente no navegador em que ela foi desenvolvida, ela pode ficar totalmente confusa em outro navegador ou em outra versão do mesmo navegador, ou pelo menos em uma resolução diferente.

Uma alternativa melhor é usar folhas de estilo. As folhas de estilo nos editores de textos permitem que os autores associem o texto a um estilo lógico, em vez de um estilo físico, por exemplo, 'parágrafo inicial' em vez de 'texto em itálico'. A aparência de cada estilo é definida separadamente. Dessa forma, se o autor decidir mudar os parágrafos iniciais de itálico em azul com corpo 14 para negrito em pink no corpo 18, basta mudar uma definição para converter o documento inteiro.

Com **CSS** (**Cascading Style Sheets**) introduziram-se folhas de estilo na Web com a HTML 4.0, embora o uso generalizado e o suporte do navegador não fossem comuns antes do ano 2000. CSS define uma linguagem simples para descrever regras que controlam a aparência do conteúdo marcado. Vejamos um exemplo. Suponha que a AWI queira usar páginas enfeitadas, com texto em azul-marinho (*navy*) na fonte Arial sobre um fundo claro, e níveis de cabeçalho que sejam 100 por cento e 50 por cento maiores do que o texto para cada nível, respectivamente. A definição da CSS no Quadro 7.7 representa essas regras.

```
body {background-color:linen; color:navy; font-family:Arial;}
h1 {font-size:200%;}
h2 {font-size:150%;}
```

Quadro 7.7 | Exemplo de CSS.

Como podemos ver, as definições de estilo podem ser compactas. Cada linha seleciona um elemento ao qual uma definição de estilo será aplicada e oferece o valor das propriedades. As propriedades de um elemento se aplicam como padrão para todos os outros elementos HTML que ele contém. Assim, o estilo para body define o estilo para os parágrafos de texto no corpo. Também existem formas abreviadas convenientes para os nomes de cor (por exemplo, red, vermelha). Quaisquer parâmetros de estilo que não sejam definidos são preenchidos com os padrões do navegador. Esse comportamento torna as definições de folha de estilo opcionais; haverá alguma apresentação razoável sem elas.

As folhas de estilo podem ser colocadas em um arquivo HTML (por exemplo, usando a tag <style>), porém é mais comum colocá-las em um arquivo separado e referenciá-las. Por exemplo, a tag <head> da página da AWI pode ser modificada para se referir a uma folha de estilo no arquivo *awistyle.css*, como mostra o Quadro 7.8. O exemplo também mostra o tipo MIME dos arquivos CSS como sendo *text/css*.

```
<head>
<title> AMALGAMATED WIDGET, INC. </title>
<link rel = "stylesheet" type = "text/css" href = "awistyle.css" />
</head>
```

Quadro 7.8 | Incluindo uma folha de estilo CSS.

Tal estratégia tem duas vantagens. Primeiro, ela permite que um conjunto de estilos seja aplicado a muitas páginas em um site. Essa organização gera uma aparência coerente às páginas, mesmo que elas sejam desenvolvidas por diferentes autores em diferentes ocasiões, e permite que a aparência do site inteiro seja mudada editando um arquivo CSS, e não a HTML. Esse método pode ser comparado com um arquivo #include em um programa em C — mudar ali uma definição de uma macro faz mudar todos os arquivos

de programa que incluem o cabeçalho. A segunda vantagem é que os arquivos HTML que são baixados se tornam menores. Isso porque o navegador pode baixar uma cópia do arquivo CSS para todas as páginas que o referenciam. Ele não precisa baixar uma nova cópia das definições junto com cada página Web.

7.3.3 Páginas Web dinâmicas e aplicações Web

O modelo de página estática que usamos até aqui trata as páginas como documentos de multimídia, que são convenientemente interligados. Esse foi um modelo útil nos primeiros dias da Web, quando grandes quantidades de informação eram colocadas on-line. Atualmente, grande parte do entusiasmo em torno da Web é seu uso para aplicações e serviços. Alguns exemplos incluem a compra de produtos em sites de comércio eletrônico, pesquisa em catálogos de biblioteca, exibição de mapas, leitura e envio de e-mail e colaboração em documentos.

Esses novos usos são como o software de aplicação tradicional (por exemplo, leitores de correio e processadores de textos). A diferença é que essas aplicações são executadas dentro do navegador, com os dados do usuário armazenados em servidores nos centros de dados da Internet. Elas usam protocolos Web para acessar informações, e o navegador para exibir uma interface com o usuário. A vantagem dessa técnica é que os usuários não precisam instalar programas de aplicação separados e os dados do usuário podem ser acessados a partir de diferentes computadores e apoiados pelo operador do serviço. Ela provou ser tão bem-sucedida que está competindo com o software de aplicação tradicional. Naturalmente, o fato de que essas aplicações são oferecidas gratuitamente por grandes provedores ajuda. Esse modelo é a forma prevalente de **computação em nuvem**, em que a computação passa dos computadores de desktop individuais para clusters de servidores compartilhados na Internet.

Para atuar como aplicações, as páginas Web não podem mais ser estáticas. Um conteúdo dinâmico é necessário. Por exemplo, uma página do catálogo da biblioteca deve refletir quais livros estão disponíveis atualmente e quais estão fora e, portanto, não disponíveis. De modo semelhante, uma página útil do mercado de ações permitiria ao usuário interagir com a página para ver preços de ações por diferentes períodos de tempo e calcular lucros e perdas. Como esses exemplos sugerem, o conteúdo dinâmico pode ser gerado por programas que são executados no servidor ou no navegador (ou nos dois lugares).

Nesta seção, examinaremos cada um desses casos, um de cada vez. A situação é aquela mostrada na Figura 7.10. Por exemplo, considere um serviço de mapa que permite ao usuário inserir um endereço e apresenta um mapa correspondente do local. Dada uma solicitação por um local, o servidor Web precisa usar um programa para criar uma página que mostre o mapa para o local a partir de um banco de dados de ruas e outras informações geográficas. Essa ação aparece como as etapas de 1 a 3. A solicitação (etapa 1) faz com que um programa seja executado no servidor. O programa consulta um banco de dados para gerar a página apropriada (etapa 2) e a retorna ao navegador (etapa 3).

Entretanto, o conteúdo não é só isso. A página é retornada por conter programas que são executados no navegador. Em nosso exemplo de mapa, o programa permitiria ao usuário encontrar rotas e explorar as áreas vizinhas em diferentes níveis de detalhe. Ele atualizaria a página, aproximaria ou afastaria a imagem conforme instruído pelo usuário (etapa 4). Para lidar com algumas interações, o programa pode precisar de mais dados do servidor. Nesse caso, ele enviará uma solicitação ao servidor (etapa 5), que capturará mais informações do banco de dados (etapa 6) e retornará uma resposta (etapa 7). O programa, então, continuará atualizando a página (etapa 4). As solicitações e respostas acontecem em segundo plano; o usuário pode nem sequer saber delas, pois o URL da página e o título normalmente não mudam. Incluindo programas no lado do cliente, a página pode apresentar uma interface mais responsiva do que apenas com os programas no lado do servidor.

Geração dinâmica de páginas Web do lado servidor

Vejamos o caso da geração de conteúdo no lado do servidor com mais detalhes. Uma situação simples, em que o processamento no lado do servidor é necessário, é o uso de formulários. Pense em um usuário preenchendo o formulário da AWI mostrado no Quadro 7.5(b) e clicando no botão *Submit order* (enviar pedido). Quando o usuário clica no botão, uma solicitação parte para o servidor no URL es-

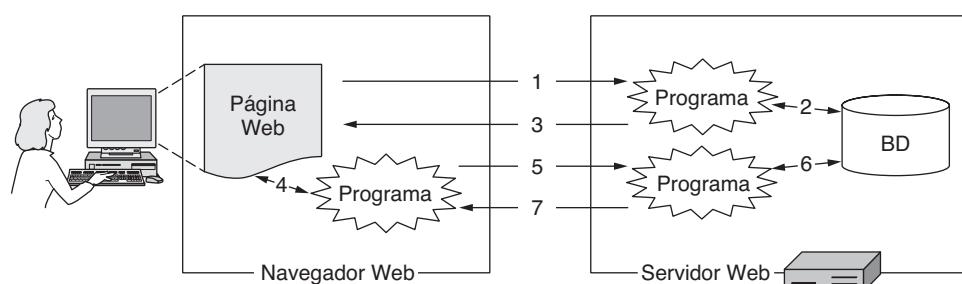



Figura 7.10 | Páginas dinâmicas.

pecificado com o formulário (um *POST* para http://widget.com/cgi-bin/order.cgi, neste caso), junto com o conteúdo do formulário conforme preenchido pelo usuário. Esses dados precisam ser emtregues a um programa ou script para que sejam processados. Assim, o URL identifica o programa a ser executado; os dados são fornecidos ao programa como entrada. Nesse caso, o processamento envolveria entrar com o pedido no sistema interno da AWI, atualizando registros de clientes e cobrando do cartão de crédito. A página retornada por essa solicitação dependerá do que acontece durante o processamento. Ela não é fixa, como uma página estática. Se o pedido tiver sucesso, a página retornada poderia informar a data prevista para a entrega. Se não tiver sucesso, a página retornada poderia informar que os produtos solicitados estão indisponíveis ou que o cartão de crédito não foi aceito por algum motivo.

O modo exato como o servidor executa um programa em vez de recuperar um arquivo depende do projeto do servidor Web. Isso não é especificado pelos próprios protocolos da web. Isso porque a interface pode ser fechada e o navegador não precisa conhecer os detalhes. Do ponto de vista do navegador, ele está simplesmente fazendo uma solicitação e buscando uma página.

Apesar disso, APIs-padrão foram desenvolvidas para os servidores Web chamarem os programas. A existência dessas interfaces torna mais fácil para os desenvolvedores estenderem diferentes servidores com aplicações Web. Veremos rapidamente duas APIs para lhe dar uma ideia do que elas envolvem.

A primeira API é um método para lidar com solicitações de página dinâmicas, que estão disponíveis desde o início da Web. Ele é chamado **CGI** (**Common Gateway Interface**) e está definido na RFC 3875. CGI oferece uma interface para permitir que servidores Web falem com programas de back-end e scripts que podem aceitar entrada (por exemplo, de formulários) e gerem páginas HTML em resposta. Esses programas podem ser escritos em qualquer linguagem conveniente para o desenvolvedor, normalmente uma linguagem de scripting para facilitar o desenvolvimento. Escolha entre Python, Ruby, Perl ou sua linguagem favorita.

Por convenção, os programas chamados via CGI residem em um diretório chamado *cgi-bin*, visível no URL. O servidor mapeia uma solicitação para esse diretório a um nome de programa e executa esse programa como um processo separado. Ele fornece quaisquer dados enviados com a solicitação como entrada para o programa. A saída do programa fornece uma página Web que é retornada ao navegador.

Em nosso exemplo, o programa *order.cgi* é chamado com a entrada do formulário, codificada como mostra o Quadro 7.6. Ele analisará os parâmetros e processará o pedido. Uma convenção útil é que o programa retorne a HTML para o formulário de pedido se nenhuma entrada do formulário estiver disponível. Desse modo, o programa estará certo de que conhece a representação do formulário.

A segunda API que examinaremos é bem diferente. A técnica aqui é embutir pequenos scripts dentro das páginas HTML e fazer com que eles sejam executados pelo próprio servidor para gerar a página. Uma linguagem popular para escrever esses scripts é **PHP** (**Hypertext Preprocessor**). Para usá-la, o servidor precisa entender PHP, assim como um navegador precisa entender CSS para interpretar as páginas Web com folhas de estilo. Normalmente, os servidores identificam páginas Web contendo PHP pela extensão de arquivo *php*, em vez de *html* ou *htm*.

PHP é mais simples de usar do que CGI. Como um exemplo de como isso funciona com formulários, veja o Quadro 7.9(a). A parte superior dessa figura contém uma página HTML normal, com um formulário simples. Dessa vez, a tag <form> especifica que *action.php* deve ser chamado para tratar dos parâmetros quando o usuário enviar o formulário. A página apresenta duas caixas de texto, uma com uma solicitação de um nome e outra com uma solicitação de uma idade. Depois que as duas caixas tiverem sido preenchidas e o formulário for enviado, o servidor analisa a string do tipo do Quadro 7.6 enviada de volta, colocando o nome na variável *name* e a idade na variável *age*. Depois, ele começa a processar o arquivo *action.php*, mostrado no Quadro 7.9(b), como resposta. Durante o processamento desse arquivo, os comandos PHP são executados. Se o usuário preencheu 'Barbara' e '32' nas caixas, o arquivo HTML enviado de volta será aquele indicado no Quadro 7.9(c). Assim, o tratamento de formulários torna-se extremamente simples usando PHP.

Embora PHP seja fácil de usar, na realidade, trata-se de uma linguagem de programação poderosa, para a interface entre a Web e um banco de dados do servidor. Ela contém variáveis, strings, arrays e a maior parte das estruturas de controle encontradas em C, mas um sistema de E/S muito mais poderoso do que apenas *printf*. PHP tem código fonte aberto, está disponível gratuitamente e é uma linguagem muito utilizada. Ela foi projetada especificamente para funcionar bem com o Apache, que também tem código fonte aberto e é o servidor Web mais utilizado no mundo. Para obter mais informações sobre PHP, consulte Valade (2009).

Até aqui, já vimos duas maneiras de gerar páginas HTML dinâmicas: scripts CGI e PHP inserida. Existem várias outras à escolha. **JSP** (**JavaServer Pages**) é semelhante a PHP, mas a parte dinâmica é escrita na linguagem de programação Java, em vez de PHP. As páginas que usam essa técnica possuem a extensão de arquivo *jsp*. **ASP.NET** (**Active Server Pages .NET**) é a versão da Microsoft para PHP e JSP. Ela usa programas escritos no framework de aplicação em rede .NET da própria Microsoft para gerar o conteúdo dinâmico. As páginas que usam essa técnica possuem a extensão *aspx*. A escolha entre essas três técnicas normalmente tem mais a ver com política (aberto ou Microsoft) do que com a tecnologia, pois elas possuem recursos semelhantes.

```
<html>
<body>
<form action = "action.php" method = "post">
<p> Please enter your name: <input type = "text" name = "name"> </p>
<p> Please enter your age: <input type = "text" name = "age"> </p> <input type = "submit">
</form>
</body>
</html>
```

(a)

```
<html>
<body>
<h1> Reply: </h1>
Hello <?php echo $name; ?>.
Prediction: next year you will be <?php echo $age + 1; ?>
</body>
</html>
```

(b)

```
<html>
<body>
<h1> Reply: </h1>
Hello Barbara.
Prediction: next year you will be 33
</body>
</html>
```

(c)

Quadro 7.9 | (a) Uma página Web contendo um formulário. (b) Um script PHP para o tratamento da saída do formulário. (c) Saída do script PHP quando as entradas são 'Barbara' e '32', respectivamente.

Geração dinâmica de páginas Web do lado cliente

Os scripts PHP e CGI resolvem o problema de manipular a entrada e as interações com bancos de dados no servidor. Todos eles podem aceitar informações de entrada de formulários, pesquisar informações em um ou mais bancos de dados e gerar páginas HTML com os resultados. O que nenhum deles pode fazer é responder a movimentos do mouse ou interagir diretamente com os usuários. Para esse propósito, é necessário ter scripts incorporados em páginas HTML executadas na máquina cliente e não na máquina servidora. A partir da HTML 4.0, esses scripts são permitidos com o uso da tag <script>. As tecnologias usadas para produzir essas páginas interativas são conhecidas de modo geral como **HTML dinâmica**.

A linguagem de scripting mais popular para o lado do cliente é o **JavaScript**, de modo que iremos examiná-la rapidamente agora. Apesar das semelhanças nos nomes, JavaScript não tem quase nada a ver com a linguagem de programação Java. Como outras linguagens de scripting, essa é uma linguagem de nível muito alto. Por exemplo, em uma única linha de JavaScript é possível mostrar uma caixa de diálogo, aguardar a entrada de texto e armazenar a string resultante em uma variável. Características de alto nível como essa tornam o JavaScript ideal para projetar páginas Web interativas. Por outro lado, o fato de não ser padronizada e de mudar com muita rapidez torna extremamente difícil escrever programas em JavaScript que funcionem em todas as plataformas, mas talvez algum dia essa linguagem se estabilize.

Como exemplo de um programa em JavaScript, considere o código do Quadro 7.10. Como o do Quadro 7.9, ele exibe um formulário que pede o nome e a idade de uma pessoa, e depois prevê qual será a idade da pessoa no ano seguinte. O corpo é quase igual ao do exemplo em PHP; a principal diferença é a declaração do botão de envio e a instrução de atribuição que ele contém. Essa instrução de atribuição informa que o navegador deve chamar o script *response* quando houver um clique em um botão e repassá-lo ao formulário como um parâmetro.

```
<html>
<head>
<script language = "javascript" type = "text/javascript">
function response(test#form) {
      var person  test#form.name.value;
      var years  eval(test#form.age.value) + 1;
      document.open();
      document.writeln("<html> <body>");
      document.writeln("Hello " + person + ".br");
      document.writeln("Prediction: next year you will be " + years + ".");
      document.writeln("</body> </html>");
      document.close();
}
</script>
</head>

<body>
<form>
Please enter your name: <input type = "text" name = "name">
<p>
Please enter your age: <input type = "text" name = "age">
<p>
<input type = "button" value = "submit" onclick = "response(this.form)">
</form>
</body>
/html
```

Quadro 7.10 | Uso de JavaScript para processar um formulário.

A grande novidade aqui é a declaração da função JavaScript *response* no início do arquivo HTML, uma área normalmente reservada para títulos, cores de fundo e assim por diante. Essa função extrai o valor do campo *name* do formulário e o armazena na variável *person*, sob a forma de uma string. Ela também extrai o valor do campo *age*, converte esse valor em um inteiro usando a função *eval*, soma 1 ao valor e armazena o resultado em *years*. Em seguida, ela abre um documento para saída, executa quatro operações de escrita usando o método *writeln* e fecha o documento. O documento é um arquivo HTML, como podemos observar pelas diversas tags HTML que ele contém. O navegador exibe então o documento na tela.

É muito importante compreender que, embora PHP e JavaScript pareçam semelhantes por ambas as linguagens incorporarem código em arquivos HTML, elas são processadas de forma totalmente diferente. No exemplo PHP do Quadro 7.9, depois que o usuário clica no botão de envio, o navegador reúne as informações em uma longa string e a envia ao servidor como solicitação para uma página PHP. O servidor carrega o arquivo PHP e executa o script PHP que está inserido, a fim de produzir uma nova página HTML. Essa página é enviada de volta ao navegador para exibição. O navegador não sabe sequer que ela foi produzida por um programa. Esse processamento aparece nas etapas de 1 a 4 da Figura 7.11(a).

No exemplo JavaScript do Quadro 7.10, quando o botão de envio é acionado, o navegador interpreta uma função JavaScript contida na página. Todo o trabalho é feito no local, dentro do navegador. Não há nenhum contato com o servidor. Esse processamento é mostrado como as etapas 1 e 2 da Figura 7.11(b). Em consequência disso, o resultado é exibido quase instantaneamente; no caso do script PHP, pode haver um atraso de vários segundos antes de o código HTML resultante chegar ao cliente.

Essa diferença não significa que o JavaScript seja melhor que o PHP. Seus usos são completamente diferentes. O script PHP (e, por consequência, JSP e ASP) é usado quando é necessária a interação com um banco de dados no servidor. JavaScript (e outras linguagens no lado do cliente que mencionaremos, como VBScript) é utilizado quando a interação se dá com o usuário no computador cliente. Sem dúvida é possível combiná-los, como veremos em breve.

JavaScript não é a única maneira de tornar as páginas Web altamente interativas. Uma alternativa em plataformas Windows é o **VBScript**, que é baseado no Visual Basic. Ou-

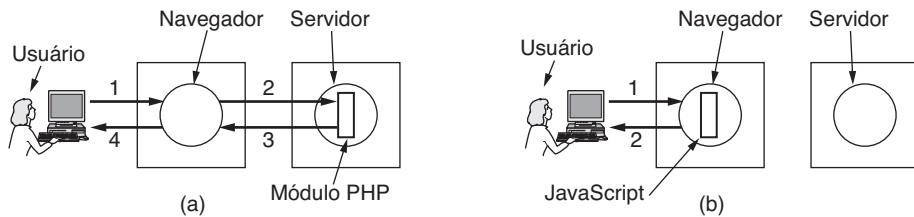



Figura 7.11 | (a) Scripting no lado do servidor com PHP. (b) Scripting no lado do cliente com JavaScript.

tro método popular entre as plataformas é usar miniaplicativos (**applets**). Estes são pequenos programas em Java que foram compilados em instruções de máquina para um computador virtual chamado **JVM (Java Virtual Machine)**. Os miniaplicativos podem ser incorporados em páginas HTML (entre as tags <applet> e </applet>) e interpretados por navegadores compatíveis com JVM. Pelo fato de os miniaplicativos Java serem interpretados em vez de executados diretamente, o interpretador Java pode evitar que eles se comportem mal, pelo menos em teoria. Na prática, os criadores de miniaplicativos descobriram uma coleção quase infinita de bugs a ser explorados nas bibliotecas de E/S Java.

A resposta da Microsoft aos miniaplicativos Java da Sun foi permitir que as páginas Web contivessem **controles ActiveX**, que são programas compilados na linguagem de máquina do x86 e executados no hardware bruto. Esse recurso os torna imensamente mais rápidos e mais flexíveis que os miniaplicativos Java interpretados, porque eles podem fazer tudo o que um programa é capaz de realizar. Quando o Internet Explorer detecta um controle ActiveX em uma página Web, ele o transfere por download, verifica sua identidade e o executa. Porém, a transferência e a execução de programas externos faz surgir questões de segurança importantes, como veremos no Capítulo 8.

Tendo em vista que quase todos os navegadores podem interpretar tanto programas em Java quanto em JavaScript, um projetista que queira criar uma página altamente interativa tem a opção de usar pelo menos duas técnicas e, se a portabilidade para várias plataformas não for um problema, ele ainda terá o ActiveX à sua disposição. Como regra geral, os programas JavaScript são mais fáceis de escrever, os miniaplicativos Java são executados com mais rapidez, e os controles ActiveX são os mais rápidos de todos. Além disso, como todos os navegadores implementam exatamente a mesma JVM, mas não há dois navegadores que implementem a mesma versão de JavaScript, os miniaplicativos Java são mais portáveis que os programas em JavaScript. Para obter mais informações sobre o JavaScript, existem muitos livros, cada um com muitas páginas (frequentemente, mais de 1.000). Veja, por exemplo, Flanagan (2010).

AJAX — Asynchronous JavaScript and XML

Aplicações Web atraentes precisam de interfaces responsivas com o usuário, acessando de forma transparente os dados armazenados em servidores Web remotos. O scripting no cliente (por exemplo, com JavaScript) e o servidor (por exemplo, com PHP) são tecnologias básicas, que oferecem partes da solução. Essas tecnologias em geral são usadas com outras tecnologias importantes, em uma combinação chamada **AJAX** (**Asynchronous JAvascript and XML**). Muitas aplicações Web completas, como o Gmail, Maps e Docs da Google, são escritas com AJAX.

AJAX é um tanto confusa porque não é uma linguagem, mas um conjunto de tecnologias que trabalham juntas para habilitar aplicações Web que são tão responsivas e poderosas quanto aplicações de desktop tradicionais. As tecnologias são:

1. HTML e CSS para apresentar informações como páginas;
2. DOM (Document Object Model) para alterar partes das páginas enquanto elas são exibidas;
3. XML (eXtensible Markup Language) para permitir que os programas troquem dados da aplicação com o servidor;
4. Um modo assíncrono para os programas enviarem e receberem dados XML;
5. JavaScript como uma linguagem para juntar toda essa funcionalidade.

Como essa é uma coleção muito grande, vamos examinar cada parte para ver com o que ela contribui. Já vimos HTML e CSS. Eles são padrões para descrever o conteúdo e como ele deve ser exibido. Qualquer programa que possa produzir HTML e CSS pode usar um navegador web como mecanismo de exibição.

DOM (**Document Object Model**) é uma representação de uma página HTML que é acessível aos programas. Essa representação é estruturada como uma árvore que reflete a estrutura dos elementos HTML. Por exemplo, a árvore DOM HTML no Quadro 7.9(a) aparece na Figura 7.12. Na raiz está um elemento *html* que representa o bloco HTML inteiro. Esse elemento é o pai do elemento *body*, que por sua vez é pai de um elemento *form*. O formulário tem dois atributos desenhados no lado direito: um para o método do formulário (um *POST*) e um para a ação do formulário (o URL a solicitar). Esse elemento tem três filhos, refletindo as duas tags de parágrafo e uma tag de entrada que estão contidas dentro do formulário. Na parte inferior da árvore estão as folhas que contêm elementos ou literais, como strings de texto.

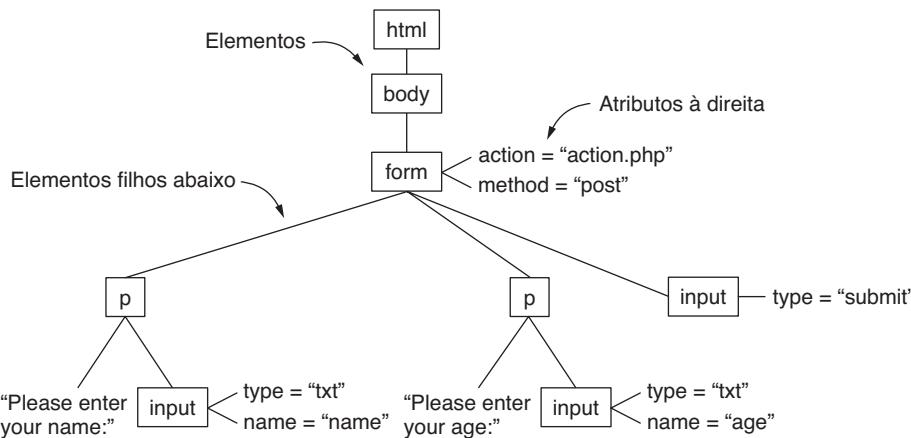



Figura 7.12 | A árvore DOM HTML do Quadro 7.9(a).

A significância do modelo DOM é que ele oferece aos programas um modo simples de alterar partes da página. Não é preciso reescrever a página inteira. Somente o nó que contém a mudança precisa ser substituído. Quando essa mudança é feita, o navegador atualizará a tela de modo correspondente. Por exemplo, se uma imagem em parte da página for alterada em DOM, o navegador atualizará essa imagem sem mudar as outras partes. Já vimos DOM em ação quando o exemplo em JavaScript do Quadro 7.10 acrescentou linhas no elemento *document* para fazer com que novas linhas de texto aparecessem no final da janela do navegador. DOM é um método poderoso para produzir páginas que podem evoluir.

A terceira tecnologia, **XML** (**eXtensible Markup Language**), é uma linguagem para especificar conteúdo estruturado. HTML mistura conteúdo com formatação, pois trata da apresentação da informação. Porém, à medida que aplicações Web se tornam mais comuns, existe uma necessidade cada vez maior de separar o conteúdo estruturado de sua apresentação. Por exemplo, considere um programa que pesquisa na Web o melhor preço para algum livro. Ele precisa analisar muitas páginas, procurando o título e o preço do item. Com páginas em HTML, é muito difícil que um programa descubra onde está o título e onde está o preço.

Por esse motivo, o W3C desenvolveu a XML (Bray et al., 2006) para permitir que o conteúdo Web fosse estruturado para processamento automatizado. Diferentemente da HTML, não existem tags definidas para XML. Cada usuário pode definir suas próprias tags. Um exemplo simples de um documento XML aparece no Quadro 7.11. Ele define uma estrutura chamada book_list, que é uma lista de books. Cada livro tem três campos, o título, o autor e o ano de publicação. Essas estruturas são muito simples. É permitido ter estruturas com campos repetidos (por exemplo, vários autores), campos opcionais (por exemplo, URL do material complementar do livro) e campos alternativos (por exemplo, URL de uma livraria, se ele estiver disponível, ou URL de um site de leilão, se estiver esgotado).

Nesse exemplo, cada um dos três campos é uma entidade indivisível, mas também é permitido subdividir os campos ainda mais. Por exemplo, o campo autor poderia ter sido criado da seguinte forma, para oferecer controle mais minucioso sobre a pesquisa e formatação:

```
<?xml version = "1.0" ?>

<book_list>

<book>
  <title> Human Behavior and the Principle of Least Effort </title>
  <author> George Zipf </author>
  <year> 1949 </year>
</book>

<book>
  <title> The Mathematical Theory of Communication </title>
  <author> Claude E. Shannon </author>
  <author> Warren Weaver </author>
  <year> 1949 </year>
</book>

<book>
  <title> Nineteen Eighty-Four </title>
  <author> George Orwell </author>
  <year> 1949 </year>
</book>

</book_list>
```

Quadro 7.11 | Um documento XML simples.

```
<author>
    <first_name> George </first_name>
    <last_name> Zipf </last_name>
</author>
```

Cada campo pode ser subdividido em subcampos e 'sub-subcampos', com qualquer profundidade.

Tudo o que o arquivo do Quadro 7.11 faz é definir uma lista de livros contendo três livros. Ele é bem adequado para transportar informações entre programas sendo executados em navegadores e servidores, mas não trata de forma alguma da exibição do documento como uma página Web. Para fazer isso, um programa que consome a informação e julga que 1949 é um ano bom para livros poderia enviar uma HTML em que os títulos são marcados como texto em itálico. Como alternativa, uma linguagem chamada **XSLT** (**eXtensible Stylesheet Language Transformations**) pode ser usada para definir como a XML deve ser transformada em HTML. XSLT é como CSS, porém muito mais poderosa. Vamos poupá-lo dos detalhes.

A outra vantagem de expressar dados em XML, e não em HTML, é que isso é mais fácil para os programas analisarem. HTML foi escrita originalmente à mão (e frequentemente ainda é), de modo que grande parte dela é um tanto descuidada. Às vezes, as tags de fechamento, como </p>, são omitidas. Outras tags não possuem uma tag de fechamento, como <br>. Outras ainda podem ser aninhadas incorretamente, e o uso de maiúsculas e minúsculas nos nomes de tags e atributos pode variar. A maioria dos navegadores faz o máximo para descobrir o que provavelmente foi intencionado. A XML é mais estrita e mais limpa em sua definição. Os nomes de tag e seus atributos sempre são escritos em minúscula, as tags sempre precisam ser fechadas na ordem reversa da ordem em que foram abertas (ou indicar claramente se são uma tag vazia, sem um fechamento correspondente) e os valores de atributo precisam ser delimitados entre aspas. Essa precisão torna a análise mais fácil e não causa ambiguidades.

A HTML está até mesmo sendo definida em termos de XML. Essa técnica é chamada **XHTML (eXtended HyperText Markup Language)**. Basicamente, essa é uma versão muito exigente da HTML. Páginas XHTML precisam estar em conformidade estrita com as regras da XML, ou então não serão aceitas pelo navegador. Chega de páginas Web de má qualidade e inconsistências entre os navegadores. Assim como XML, a intenção é produzir páginas que sejam melhores para os programas (neste caso, as aplicações Web) processarem. Embora XHTML já exista desde 1998, sua utilização em geral tem sido lenta. As pessoas que produzem HTML não veem motivo para usar XHTML, e o suporte do navegador tem sido lento. Agora, a HTML 5.0 está sendo definida de modo que uma página possa ser representada como HTML ou XHTML, para ajudar na transição. No futuro, XHTML deverá substituir HTML, mas ainda haverá muito tempo antes que essa transição esteja completa.

A XML também provou ser popular como uma linguagem para a comunicação entre programas. Quando essa comunicação é executada pelo protocolo HTTP (descrito na próxima seção), ela é chamada de Web service. Em particular, **SOAP** (**Simple Object Access Protocol**) é uma maneira de implementar Web services que realizam RPC entre programas de uma maneira independente de linguagem e sistema. O cliente simplesmente constrói a solicitação como uma mensagem XML e a envia ao servidor, usando o protocolo HTTP. O servidor devolve uma resposta como uma mensagem formatada em XML. Desse modo, as aplicações em plataformas heterogêneas podem se comunicar.

Voltando ao AJAX, nossa conclusão é simplesmente que XML é um formato útil para a troca de dados entre os programas rodando no navegador e no servidor. Porém, para fornecer uma interface responsiva no navegador enquanto os dados são enviados ou recebidos, é possível que os scripts realizem **E/S assíncrona**, o que não impede a exibição enquanto se espera pela resposta a uma solicitação. Por exemplo, considere um mapa que pode ser rolado no navegador. Quando notificado da ação de rolagem, o script na página do mapa pode solicitar mais dados do mapa do servidor, se a visão do mapa estiver próxima da borda dos dados. A interface não deverá ser congelada enquanto os dados são buscados. Essa interface não agradaria aos usuários. Em vez disso, a rolagem deverá continuar tranquilamente. Quando os dados chegarem, o script será notificado para poder usar os dados. Se tudo correr bem, novos dados do mapa serão buscados antes de ser necessários. Os navegadores modernos admitem esse modelo de comunicação.

A última peça do quebra-cabeça é uma linguagem de scripting que reúna o AJAX, oferecendo acesso à lista de tecnologias apresentada. Quase sempre, essa linguagem é JavaScript, mas existem alternativas, como VBScript. Já mostramos um exemplo simples de JavaScript. Não se engane com essa simplicidade. JavaScript tem muitos detalhes, mas é uma linguagem de programação completa, com todo o poder das linguagens C ou Java. Ela tem variáveis, strings, arrays, objetos, funções e todas as estruturas de controle normais. Ela também possui interfaces específicas para o navegador e para as páginas Web. JavaScript pode acompanhar o movimento do mouse sobre objetos na tela, tornando fácil fazer com que um menu de repente apareça e leve a páginas Web dinâmicas. Ela pode usar DOM para acessar páginas, manipular HTML e XML e realizar comunicação HTTP assíncrona.

Antes de sairmos do assunto de páginas dinâmicas, vamos resumir rapidamente as tecnologias que vimos até aqui, relacionando-as em uma única figura. Páginas completas podem ser geradas instantaneamente por vários scripts na máquina servidora. Os scripts podem ser escritos em linguagens de extensão de servidor, como PHP, JSP ou ASP.NET, ou executados como processos CGI separados e, portanto, ser escritos em qualquer linguagem. Essas opções aparecem na Figura 7.13.

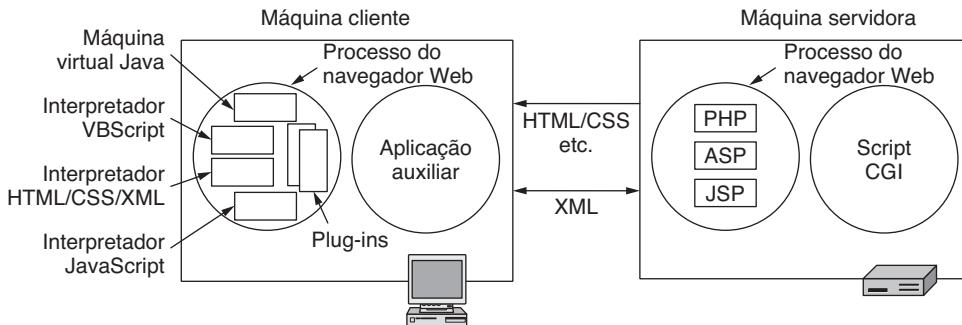



Figura 7.13 | Diversas tecnologias usadas para gerar páginas dinâmicas.

Quando essas páginas Web são recebidas pelo navegador, elas são tratadas como páginas normais em HTML, CSS e outros tipos MIME e são simplesmente exibidas. Os plug-ins que rodam no navegador e as aplicações auxiliares que rodam fora do navegador podem ser instalados para estender os tipos MIME que são admitidos pelo navegador.

A geração dinâmica de conteúdo também é possível no lado do cliente. Os programas que estão inseridos nas páginas Web podem ser escritos em JavaScript, VBScript, Java e outras linguagens. Esses programas podem realizar computação e atualizar a tela. Com AJAX, os programas nas páginas podem trocar, de modo assíncrono, XML e outros tipos de dados com o servidor. Esse modelo admite aplicações Web ricas, que se parecem com aplicações tradicionais, mas rodam dentro do navegador e acessam informações que estão armazenadas nos servidores na Internet.

7.3.4 HTTP — HyperText Transfer Protocol

Agora que já entendemos o conteúdo e as aplicações da Web, é hora de examinarmos o protocolo que é usado para transportar toda essa informação entre os servidores e os clientes na Web. Ele é o **HTTP** (**HyperText Transfer Protocol**), especificado na RFC 2616.

HTTP é um protocolo simples, do tipo solicitação-resposta, que roda sobre TCP. Ele especifica quais mensagens os clientes podem enviar para os servidores e quais respostas recebem de volta. Os cabeçalhos de solicitação e resposta são dados em ASCII, assim como no SMTP. O conteúdo é dado em formato tipo MIME, também como no SMTP. Esse modelo simples foi em parte responsável pelo sucesso inicial da Web, pois simplificou o desenvolvimento e a implantação.

Nesta seção, examinaremos as propriedades mais importantes do HTTP, conforme ele é usado atualmente. Porém, antes de entrar nos detalhes, temos que observar que o modo como ele é usado na Internet está em evolução. O HTTP é um protocolo da camada de aplicação, pois roda em cima do TCP e está bastante associado à Web. É por isso que ele está sendo explicado neste capítulo. Porém, em outro sentido, o HTTP está se tornando mais um protocolo de transporte, que oferece um meio para os processos comunicarem conteúdo entre os limites de diferentes redes. Esses processos não precisam ser um navegador Web nem um servidor Web. Um player de mídia poderia usar HTTP para falar com um servidor e solicitar informações do álbum. O software antivírus poderia usar HTTP para baixar as atualizações mais recentes. Os desenvolvedores poderiam usar HTTP para buscar arquivos do projeto. Produtos eletrônicos de consumo, como quadros de fotos digitais, normalmente usam um servidor HTTP inserido como interface para o mundo exterior. A comunicação entre máquinas cada vez mais utiliza HTTP. Por exemplo, o servidor de uma companhia aérea poderia usar SOAP (um RPC XML sobre HTTP) para entrar em contato com um servidor de aluguel de carros e fazer a reserva de um carro, tudo como parte de um pacote de férias. Essas tendências provavelmente continuarão, com o uso em expansão do HTTP.

Conexões

O modo habitual de um navegador entrar em contato com um servidor é estabelecer uma conexão TCP para a porta 80 da máquina servidora, embora esse procedimento não seja exigido formalmente. A vantagem de usar o TCP é que nem os navegadores nem os servidores têm de se preocupar com mensagens longas, confiabilidade ou controle de congestionamento. Todos esses assuntos são tratados pela implementação do TCP.

Antigamente na Web, com o HTTP 1.0, depois que a conexão era estabelecida, uma única solicitação era enviada e uma única resposta era devolvida. Então, a conexão TCP era encerrada. Em um mundo no qual as páginas Web típicas consistiam inteiramente em texto HTML, esse método era adequado. Rapidamente, uma página comum continha grandes números de links inseridos para conteúdo como ícones e outros atrativos visuais. O estabelecimento de uma conexão TCP separada para transportar um único ícone se tornou um modo de operação muito dispendioso.

Essa observação levou ao lançamento do HTTP 1.1, que admite **conexões persistentes**. Com elas, é possível estabelecer uma conexão TCP, enviar uma solicitação e obter uma resposta, e depois enviar solicitações adicionais e receber respostas adicionais. Essa estratégia também é chamada **reúso**

de conexão. Amortizando os custos da instalação, início e liberação do TCP por várias solicitações, o overhead relativo devido ao TCP é muito menor por solicitação. Também é possível transportar as solicitações por pipeline, ou seja, enviar a solicitação 2 antes de chegar a resposta à solicitação 1.

A diferença de desempenho entre esses três casos aparece na Figura 7.14. A parte (a) mostra três solicitações, uma após a outra, e cada uma em uma conexão separada. Vamos supor que isso represente uma página Web com duas imagens inseridas no mesmo servidor. Os URLs das imagens são determinados à medida que a página principal é buscada, de modo que eles são buscados após a página principal. Atualmente, uma página típica tem cerca de 40 outros objetos que precisam ser buscados para apresentá-la, mas isso tornaria nossa figura muito grande, e por isso usaremos apenas dois objetos inseridos.

Na Figura 7.14(b), a página é buscada com uma conexão persistente. Ou seja, a conexão TCP é aberta no início, depois as mesmas três solicitações são enviadas, uma após a outra, como antes, e somente então a conexão é fechada. Observe que a busca completa é mais rápida. Existem dois motivos para o ganho de velocidade. Primeiro, não é gasto tempo no estabelecimento de conexões adicionais. Segundo, a transferência das mesmas imagens prossegue mais rapidamente. Por que isso acontece? É por causa do controle de congestionamento do TCP. No início de uma conexão, o TCP usa o procedimento de partida lenta para aumentar a vazão até que ele descubra o comportamento do caminho da rede. A consequência desse período de aquecimento é que várias conexões TCP curtas levam desproporcionalmente mais tempo para transferir informações do que uma conexão TCP mais longa.

Finalmente, na Figura 7.14(c), existe uma conexão persistente e as solicitações são feitas por pipeline. Especificamente, a segunda e a terceira solicitações são enviadas em rápida sucessão assim que uma parte suficiente da página principal tiver sido recuperada para identificar que as imagens devem ser capturadas. As respostas a essas solicitações chegam em seguida. Esse método reduz o tempo de ociosidade do servidor, de modo que melhora seu desempenho.

Contudo, as conexões persistentes têm um preço a ser pago. Um novo problema que elas geram é saber quando fechar a conexão. Uma conexão com um servidor deve permanecer aberta enquanto a página é carregada. E depois? Há uma boa chance de que o usuário clique em um link que solicite outra página do servidor. Se a conexão permanecer aberta, a próxima solicitação pode ser atendida imediatamente. Porém, não há garantias de que o cliente fará outra solicitação do servidor em pouco tempo. Na prática, clientes e servidores normalmente mantêm conexões persistentes abertas até que tenham ficado ociosos por um pequeno período (por exemplo, 60 segundos) ou até que haja um grande número de conexões abertas e algumas tenham que ser encerradas.

O leitor atento pode ter notado que existe uma combinação que omitimos até aqui. Também é possível enviar uma solicitação por conexão TCP, mas executar várias conexões TCP em paralelo. Esse método de **conexão paralela** foi bastante usado pelos navegadores antes das conexões persistentes. Ele tem a mesma desvantagem das conexões sequenciais (overhead extra), porém, com desempenho muito melhor. Isso porque o estabelecimento e o aumento das conexões em paralelo escondem parte da latência. Em nosso exemplo, as conexões para as duas imagens inseridas poderiam ser estabelecidas ao mesmo tempo. Porém, executar muitas conexões TCP com o mesmo servidor é algo desencorajado. O motivo é que o TCP realiza controle de congestionamento para cada conexão independentemente. Como consequência, as conexões competem entre si,

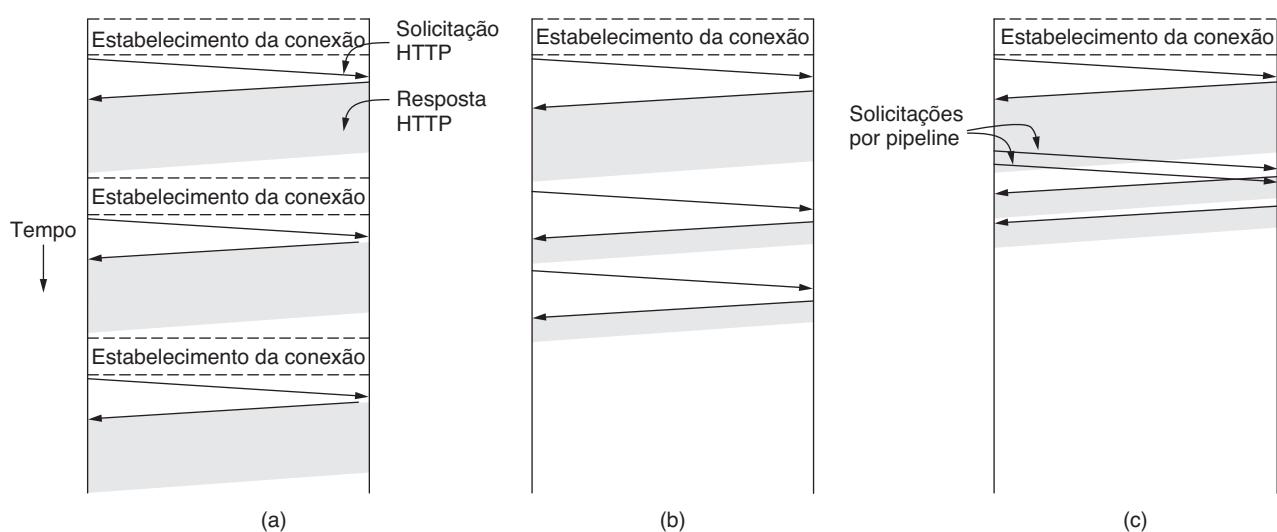



Figura 7.14 | HTTP com (a) múltiplas conexões e solicitações sequenciais. (b) Uma conexão persistente e solicitações sequenciais. (c) Uma conexão persistente e solicitações por pipeline.

causando mais perdas de pacotes e, em geral, são usuários mais agressivos da rede do que uma conexão individual. As conexões persistentes são superiores e são preferidas às conexões paralelas, pois evitam o overhead e não sofrem de problemas de congestionamento.

Métodos

Embora o HTTP tenha sido projetado para utilização na Web, ele foi criado de modo mais geral que o necessário, visando às futuras aplicações orientadas a objetos. Por essa razão, são aceitas operações chamadas **métodos**, diferentes da simples solicitação de uma página Web. Essa generalidade permitiu que o SOAP viesse a existir.

Cada solicitação consiste em uma ou mais linhas de texto ASCII, sendo a primeira palavra da primeira linha o nome do método solicitado. Os métodos internos estão listados na Tabela 7.12. Os nomes diferenciam letras maiúsculas de minúsculas; portanto, *GET* é um método válido, mas *get* não é.

O método *GET* solicita ao servidor que envie a página (ou objeto, no caso mais genérico; mas pensar em uma página como o conteúdo de um arquivo é suficiente para entender os conceitos). A página é codificada em MIME de forma adequada. A grande maioria das solicitações a servidores da Web tem a forma de métodos *GET*. A forma comum de *GET* é:

GET *nome-arquivo* HTTP/1.1

onde *nome-arquivo* identifica a página a ser buscada e 1.1 é a versão do protocolo que está sendo usado.

O método *HEAD* solicita apenas o cabeçalho da mensagem, sem a página propriamente dita. Esse método pode ser usado para reunir informações destinadas à indexação ou apenas para testar a validade de um URL.

O método *POST* é usado quando os formulários são enviados. Tanto ele quanto *GET* são usados também para Web services SOAP. Assim como *GET*, ele contém um URL, mas, em vez de simplesmente capturar uma página, ele faz o upload de dados para o servidor (ou seja, o conteúdo do formulário ou parâmetros de RPC). O servidor, então, faz algo com os dados que depende do URL, conceitualmente acrescentando os dados ao objeto. O efeito poderia ser comprar um item, por exemplo, ou chamar um procedimento. Finalmente, o método retorna uma página indicando o resultado.

Os métodos restantes não são muito usados para navegação na Web. O método *PUT* é o contrário de *GET:* em vez de ler a página, ele escreve a página. Esse método possibilita criar uma coleção de páginas Web em um servidor remoto. O corpo da solicitação contém a página. Ela pode ser codificada usando MIME, quando as linhas após o *PUT* poderiam incluir cabeçalhos de autenticação, para provar que quem chamou realmente tem permissão para realizar a operação solicitada.

DELETE faz exatamente isso: exclui a página ou, pelo menos, indica que o servidor Web concordou em excluir a página. A exemplo de *PUT*, a autenticação e a permissão têm papel fundamental.

O método *TRACE* serve para depuração. Ele instrui o servidor a enviar de volta a solicitação. Esse método é útil quando as solicitações não estão sendo processadas corretamente e o cliente deseja saber qual solicitação o servidor recebeu de fato.

O método *CONNECT* permite que um usuário faça uma conexão com um servidor Web por meio de um dispositivo intermediário, como um cache Web.

O método *OPTIONS* fornece um meio para que o cliente consulte o servidor sobre uma página e obtenha os métodos e cabeçalhos que podem ser usados com essa página.

Toda solicitação obtém uma resposta que consiste em uma linha de *status* e, possivelmente, informações adicionais (por exemplo, uma página Web ou parte dela). A linha de *status* contém um código de *status* de três dígitos informando se a solicitação foi atendida e, se não foi, por que não. O primeiro dígito é usado para dividir as respostas em cinco grupos importantes, como mostra a Tabela 7.13. Os códigos 1xx raramente são usados na prática. Os códigos 2xx significam que a solicitação foi tratada com sucesso, e que o conteúdo (se houver) está sendo retornado. Os códigos 3xx informam ao cliente que ele deve procurar em outro lugar, usando um URL diferente ou seu próprio cache (conforme descreveremos mais adiante). Os códigos 4xx significam que a solicitação falhou devido a um erro do cliente, como uma solicitação inválida ou uma página inexistente. Por fim, os erros 5xx significam que o próprio servidor tem um problema interno, seja ele causado por um erro em seu código ou por uma sobrecarga temporária.

Método	Descrição
GET	Lê uma página Web
HEAD	Lê um cabeçalho de página Web
POST	Acrescenta algo a uma página Web
PUT	Armazena uma página Web
DELETE	Remove a página Web
TRACE	Ecoa a solicitação recebida
CONNECT	Conecta através de um proxy
OPTIONS	Consulta opções para uma página

Tabela 7.12 | Os métodos internos de solicitações HTTP.

Cabeçalhos de mensagens

A linha de solicitação (por exemplo, a linha com o método *GET*) pode ser seguida por linhas adicionais com mais informações. Elas são chamadas **cabeçalhos de solicitação**. Essas informações podem ser comparadas com os parâmetros de uma chamada de procedimento. As respos-

Código	Significado	Exemplos
1xx	Informação	100 = servidor concorda em tratar da solicitação do cliente
2xx	Sucesso	200 = solicitação com sucesso; 204 = nenhum conteúdo presente
3xx	Redirecionamento	301 = página movida; 304 = página em cache ainda válida
4xx	Erro do cliente	403 = página proibida; 404 = página não localizada
5xx	Erro do servidor	500 = erro interno do servidor; 503 = tente novamente mais tarde

Tabela 7.13 | Os grupos de respostas de código de status.

tas também podem ter **cabeçalhos de resposta**. Alguns cabeçalhos podem ser usados em um ou em outro sentido. Uma seleção dos mais importantes é dada na Tabela 7.14. Essa lista não é curta, e portanto você pode imaginar que geralmente existe uma série de cabeçalhos em cada solicitação e resposta.

O cabeçalho *User-Agent* permite ao cliente informar ao servidor sobre sua implementação de navegador (por exemplo, *Mozilla/5.0* e *Chrome/5.0.375.125*). Essa informação é útil para permitir que os servidores ajustem suas respostas, pois diferentes navegadores podem ter capacidades e comportamentos variados.

Os quatro cabeçalhos *Accept* informam ao servidor o que o cliente está disposto a aceitar na eventualidade de ele ter um repertório limitado daquilo que é aceitável. O primeiro cabeçalho especifica os tipos MIME que são bem-vindos (por exemplo, *text/html*). O segundo fornece o conjunto de caracteres (por exemplo, *ISO-8859-5* ou *Unicode-1-1*). O terceiro lida com métodos de compressão (por exemplo, *gzip*). O quarto indica um idioma natural (por exemplo, português). Se o servidor tiver uma opção de páginas, ele poderá usar essas informações para fornecer o que o cliente está procurando. Se ele for incapaz de satisfazer a solicitação, será retornado um código de erro e a solicitação falhará.

Os cabeçalhos *If-Modified-Since* e *If-None-Match* são usados com o caching. Eles permitem que o cliente solicite uma página para que seja enviada somente se a cópia em cache não for mais válida. Descreveremos o caching em breve.

Host é o cabeçalho que identifica o servidor. Ele é retirado do URL. Esse cabeçalho é obrigatório, e é usado porque alguns endereços IP podem ter múltiplos nomes no DNS, e o servidor precisa ter algum meio de identificar o host a quem deve entregar a solicitação.

O cabeçalho *Authorization* é necessário para páginas protegidas. Nesse caso, o cliente talvez tenha de provar que tem direito de ver a página solicitada. Esse cabeçalho é usado para esse caso específico.

O cliente usa o cabeçalho *Referer* (que é um nome incorreto) para indicar o URL que se referiu (*referred*) ao URL que agora é solicitado. Normalmente, trata-se do URL da página anterior. Esse cabeçalho é particularmente útil para acompanhar a navegação na Web, pois informa aos servidores como um cliente chegou até a página.

Embora os cookies sejam tratados na RFC 2109 e não na RFC 2616, eles também têm cabeçalhos. O *Set-Cookie* é o modo como os servidores enviam cookies aos clientes. O cliente deverá salvar o cookie e retorná-lo em solicitações subsequentes ao servidor usando o cabeçalho *Cookie*. (Observe que existe uma especificação mais recente para os cookies, com cabeçalhos mais novos, a RFC 2965, mas ela foi em grande parte rejeitada pelo setor, e não é muito implementada.)

Muitos outros são usados nas respostas. O cabeçalho *Server* permite que o servidor identifique seu build de software, se desejar. Os cinco seguintes, todos começando com *Content-*, permitem ao servidor descrever propriedades da página que está enviando.

O cabeçalho *Last-Modified* informa quando a página foi modificada pela última vez, e o *Expires* informa por quanto tempo permanecerá válida. Esses dois cabeçalhos desempenham uma função importante no armazenamento de páginas em cache.

Location é o cabeçalho usado pelo servidor para informar ao cliente que ele deve tentar outro URL. Pode ser usado se a página tiver sido deslocada ou para permitir que vários URLs se refiram à mesma página (possivelmente em servidores distintos). Ele também é usado por empresas que têm uma página Web principal no domínio *com*, mas que redirecionam os clientes para uma página nacional ou regional de acordo com seu endereço IP ou com seu idioma preferido.

Se uma página for muito grande, um cliente pequeno talvez não queira recebê-la toda de uma vez. Alguns servidores aceitarão solicitações de intervalos de bytes, de forma que a página possa ser obtida em várias unidades pequenas. O cabeçalho *Accept-Ranges* anuncia a disposição do servidor para lidar com esse tipo de solicitação de páginas parciais.

Agora, chegamos aos que podem ser usados nos dois sentidos. O cabeçalho *Date* pode ser usado nos dois sentidos e contém a data e a hora em que a mensagem foi enviada, enquanto o *Range* informa o intervalo de bytes da página que é fornecida pela resposta.

O cabeçalho *ETag* oferece uma tag curta que serve como nome para o conteúdo da página. Ele é usado para caching. O cabeçalho *Cache-Control* indica outras instruções

Cabeçalho	Tipo	Conteúdo
User-Agent	Solicitação	Informações sobre o navegador e sua plataforma
Accept	Solicitação	O tipo de páginas que o cliente pode manipular
Accept-Charset	Solicitação	Os conjuntos de caracteres aceitáveis para o cliente
Accept-Encoding	Solicitação	As codificações de páginas que o cliente pode manipular
Accept-Language	Solicitação	Os idiomas com os quais o cliente pode lidar
If-Modified-Since	Solicitação	Data e hora para verificar atualização
If-None-Match	Solicitação	Tags enviadas anteriormente para verificar atualização
Host	Solicitação	O nome DNS do servidor
Authorization	Solicitação	Uma lista das credenciais do cliente
Referer	Solicitação	O URL anterior do qual a solicitação veio
Cookie	Solicitação	Cookie previamente definido, enviado de volta ao servidor
Set-Cookie	Resposta	Cookie para o cliente armazenar
Server	Resposta	Informações sobre o servidor
Content-Encoding	Resposta	Como o conteúdo está codificado (por exemplo, *gzip*)
Content-Language	Resposta	O idioma usado na página
Content-Length	Resposta	O tamanho da página em bytes
Content-Type	Resposta	O tipo MIME da página
Content-Range	Resposta	Identifica uma parte do conteúdo da página
Last-Modified	Resposta	Data e hora da última modificação na página
Expires	Resposta	Data e hora de quando a página deixa de ser válida
Location	Resposta	Informa para onde o cliente deve enviar sua solicitação
Accept-Ranges	Resposta	Indica que o servidor aceitará solicitações de intervalos de bytes
Date	Ambos	Data e hora em que a mensagem foi enviada
Range	Ambos	Identifica uma parte de uma página
Cache-Control	Ambos	Diretivas para o modo de tratar caches
ETag	Ambos	Tag para o conteúdo da página
Upgrade	Ambos	O protocolo para o qual o transmissor deseja passar

Tabela 7.14 | Alguns cabeçalhos de mensagem HTTP.

explícitas sobre como manter as páginas em cache (ou, como geralmente acontece, como não manter em cache).

Por fim, o cabeçalho *Upgrade* é usado para passar para um novo protocolo de comunicação, como um futuro protocolo HTTP ou um transporte seguro. Ele permite que o cliente anuncie o que ele pode aceitar e o servidor declare o que está usando.

Caching

As pessoas normalmente retornam às páginas Web que já viram antes, e páginas relacionadas quase sempre têm os mesmos recursos internos. Alguns exemplos são as imagens usadas para navegação pelo site, bem como folhas de estilo e scripts comuns. Seria um desperdício capturar todos esses recursos para as páginas toda vez que eles forem exibidos, pois o navegador já tem uma cópia.

Guardar as páginas que são buscadas para uso subsequente é chamado **caching**. A vantagem é que, quando uma página em cache pode ser reutilizada, não é necessário repetir a transferência. O HTTP possui suporte interno para ajudar os clientes a identificar quando eles podem reutilizar as páginas com segurança. Esse suporte melhora o desempenho, reduzindo tanto o tráfego quanto a latência na rede. A desvantagem é que o navegador agora precisa armazenar páginas, mas isso quase sempre compensa, pois o armazenamento local é barato. As páginas normalmente

são mantidas em disco, para que possam ser usadas quando o navegador for executado em outro momento.

A questão difícil com o caching HTTP é como determinar que uma cópia previamente mantida em cache representa a mesma página que seria exibida se fosse buscada novamente no servidor. Essa determinação não pode ser feita unicamente pelo URL. Por exemplo, o URL pode indicar uma página que apresenta o item de notícias mais recente. O conteúdo dessa página será atualizado frequentemente, embora o URL continue sendo o mesmo. Como alternativa, o conteúdo da página pode ser uma lista dos deuses da mitologia grega e romana. Essa página deverá mudar com muito menos frequência.

O HTTP usa duas estratégias para enfrentar esse problema. Elas são mostradas na Figura 7.15 como formas de processamento entre a solicitação (etapa 1) e a resposta (etapa 5). A primeira estratégia é a validação da página (etapa 2). O cache é consultado e, se tiver uma cópia de uma página para o URL solicitado, sabe-se que ela é recente (ou seja, ainda válida) e não é preciso buscá-la novamente do servidor. Em vez disso, as páginas em cache podem ser retornadas diretamente. O cabeçalho *Expires* retorna quando a página em cache foi buscada originalmente e a data e hora atuais podem ser usadas para fazer essa determinação.

Porém, nem todas as páginas vêm com um cabeçalho *Expires* conveniente, que informa quando a página deve ser buscada novamente no servidor. Afinal, é difícil fazer previsões – especialmente sobre o futuro. Nesse caso, o navegador pode usar heurísticas. Por exemplo, se a página não tiver sido modificada no ano anterior (conforme informado pelo cabeçalho *Last-Modified*), é bastante seguro apostar que não mudará na próxima hora. Porém, não há garantias, e essa pode ser uma escolha errada. Por exemplo, as ações do mercado poderiam ter fechado para o dia, de modo que a página não mudará por horas, mas ela mudará rapidamente quando a próxima sessão de negociações começar. Assim, a validade do cache de uma página pode variar muito com o tempo. Por esse motivo, as heurísticas devem ser usadas com cuidado, embora em geral funcionem bem na prática.

Encontrar páginas que não expiraram é o uso mais benéfico do caching, pois significa que o servidor não precisa ser contatato. Infelizmente, isso nem sempre funciona. Os servidores precisam usar o cabeçalho *Expires* com cautela, pois eles podem não saber com certeza quando uma página será atualizada. Assim, as cópias em cache ainda podem estar atualizadas, mas o cliente não sabe disso.

A segunda estratégia é usada nesse caso. Com ela, o servidor responde se a cópia em cache ainda é válida. Essa solicitação é um **GET condicional**, e pode ser vista na etapa 3 da Figura 7.15. Se o servidor souber que a cópia em cache ainda é válida, ele poderá enviar uma resposta curta para dizer isso (etapa 4a). Caso contrário, ele precisará enviar a resposta completa (etapa 4b).

Outros campos do cabeçalho são usados para permitir que o servidor verifique se uma cópia em cache ainda é válida. O cliente tem a hora em que uma página em cache foi atualizada pela última vez, através do cabeçalho *Last-Modified*. Ele pode enviar essa hora ao servidor usando o cabeçalho *If-Modified-Since* para solicitar a página somente se ela tiver sido alterada depois dessa hora.

Como alternativa, o servidor pode retornar um cabeçalho *ETag* com a página. Esse cabeçalho contém uma tag que é um nome curto para o conteúdo da página, como um checksum, porém melhor. (Ela pode ser um hash criptográfico, que descreveremos no Capítulo 8.) O cliente pode validar as cópias em cache enviando ao servidor um cabeçalho *If-None-Match* listando as tags das cópias em cache. Se qualquer uma das tags combinar com o conteúdo que o servidor responderia, a cópia em cache correspondente pode ser usada. Esse método pode ser usado quando não for conveniente ou útil determinar a atualização. Por exemplo, um servidor pode retornar um conteúdo diferente para o mesmo URL dependendo dos idiomas e tipos MIME preferidos. Nesse caso, a data de modificação apenas não ajudará o servidor a determinar se a página em cache está atualizada.

Por fim, observe que essas duas estratégias de caching são anuladas pelas diretivas transportadas no cabeçalho *Cache-Control*. Essas diretivas podem ser usadas para restringir o caching (por exemplo, *no-cache*) quando ele não for apropriado. Um exemplo é uma página dinâmica que será diferente da próxima vez que for capturada. As páginas que exigem autorização também não são mantidas em cache.

Há muito mais com relação ao caching, mas só temos espaço para fazer duas considerações importantes. Primeiro, ele pode ser realizado em outros lugares além do navegador. No caso geral, as solicitações HTTP podem ser roteadas por uma série de caches. O uso de um cache externo

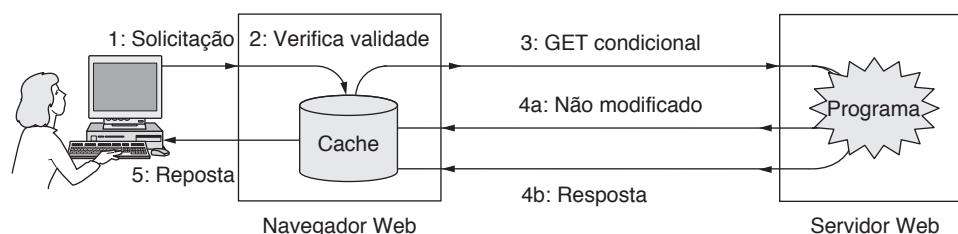



Figura 7.15 | Caching HTTP.

ao navegador é chamado **caching por proxy**. Cada nível adicional do caching pode ajudar a reduzir as solicitações mais acima na cadeia. É muito comum que organizações como ISPs e empresas executem caches por proxy para obter os benefícios do caching de páginas para diferentes usuários. Discutiremos o caching por proxy com o tópico mais amplo da distribuição de controle, na Seção 7.5, no final deste capítulo.

Segundo, os caches oferecem um aumento importante no desempenho, mas não tanto quanto se poderia esperar. O motivo é que, embora certamente existam documentos populares na Web, há também muitos documentos pouco populares que as pessoas buscam, muitos deles também muito longos (por exemplo, vídeos). A 'longa cauda' dos documentos não populares ocupa espaço nos caches, e o número de solicitações que podem ser tratadas a partir do cache cresce lentamente com seu tamanho. Os caches na Web provavelmente sempre deverão ser capazes de lidar com menos da metade das solicitações. Consulte Breslau et al. (1999) para obter mais informações.

Exemplo e utilização do HTTP

Como o HTTP é um protocolo ASCII, é muito fácil a comunicação direta entre uma pessoa em um terminal (diferente de um navegador) e servidores Web. Basta uma conexão TCP com a porta 80 no servidor. Os leitores devem experimentar pessoalmente a sequência de comandos a seguir. Ela funcionará na maioria dos shells do UNIX e na janela de comandos do Windows (desde que o programa telnet esteja habilitado).

```
telnet www.ietf.org 80
GET /rfc.html HTTP/1.1
Host: www.ietf.org
```

Essa sequência de comandos inicia uma conexão telnet (isto é, TCP) com a porta 80 no servidor Web da IETF, www.ietf.org. Em seguida, vem o comando *GET*, que identifica o caminho do URL e o protocolo. Experimente servidores e URLs à sua escolha. A próxima linha é o cabeçalho *Host* obrigatório. Uma linha em branco após o último cabeçalho também é necessária. Ela indica ao servidor que não existem mais cabeçalhos de solicitação. O servidor, em seguida, enviará a resposta. Dependendo do servidor e do URL, muitos tipos de cabeçalhos e páginas diferentes poderão ser observados.

7.3.5 A Web móvel

A Web é usada a partir de quase todo tipo de computador, e isso inclui telefones móveis. A navegação na Web por uma rede sem fios, embora móvel, pode ser muito útil. Isso também apresenta problemas técnicos, pois grande parte do conteúdo da Web foi projetado para apresentações espetaculares nos computadores de desktop, com conectividade de banda larga. Nesta seção, vamos descrever como está sendo desenvolvido o acesso à Web a partir de dispositivos móveis, ou a **Web móvel**.

Em comparação com os computadores de desktop em casa ou no trabalho, os telefones móveis apresentam várias dificuldades para a navegação na Web:

1. telas relativamente pequenas impedem páginas grandes e imagens grandes;
2. capacidades de entrada limitadas tornam cansativo digitar URLs ou outro tipo de entrada extensa;
3. a largura de banda da rede é limitada pelos enlaces sem fios, particularmente em redes de celular (3G), onde normalmente ela também é muito cara;
4. a conectividade pode ser intermitente;
5. a potência de computação é limitada, por motivos de vida da bateria, tamanho, dissipação de calor e custo.

Essas dificuldades significam que simplesmente usar conteúdo do desktop para a Web móvel provavelmente gerará uma experiência frustrante para o usuário.

As primeiras técnicas para a Web móvel criaram uma nova pilha de protocolos adaptada para dispositivos sem fios, com capacidades limitadas. **WAP** (**Wireless Application Protocol**) é o exemplo mais conhecido dessa estratégia. O esforço para o WAP foi iniciado em 1997 pelas principais companhias de telefonia móvel, como Nokia, Ericsson e Motorola. Porém, algo inesperado aconteceu ao longo do caminho. Na década seguinte, a largura de banda de rede e as capacidades de dispositivo aumentaram tremendamente com a implantação dos serviços de dados 3G e telefones móveis com telas maiores, processadores mais rápidos e capacidades de rede sem fios 802.11. De repente, foi possível que dispositivos móveis executassem navegadores da Web simples. Ainda existe uma lacuna entre esses dispositivos e os desktops, que nunca será fechada, mas muitos dos problemas de tecnologia que incentivaram uma pilha separada de protocolos desapareceram.

A técnica que cada vez é mais utilizada é usar os mesmos protocolos da Web para dispositivos móveis e de desktop, fazendo com que os sites entreguem conteúdo adaptado para o dispositivo móvel quando o usuário está em tal dispositivo. Os servidores Web são capazes de detectar se devem retornar versões das páginas Web para desktop ou dispositivo móvel examinando os cabeçalhos de solicitação. Um cabeçalho *User-Agent* é útil especialmente nesse aspecto, pois identifica o software navegador. Assim, quando um servidor Web recebe uma solicitação, ele pode examinar os cabeçalhos e retornar uma página com imagens pequenas, menos texto e navegação mais simples para um iPhone e uma página com todos os recursos para um usuário em um notebook.

O W3C tem encorajado essa técnica de várias formas. Uma delas é padronizar as melhores práticas para o conteúdo Web móvel. Uma lista de 60 dessas melhores práticas é

fornecida na primeira especificação (Rabin e McCathieNevile, 2008). A maioria dessas práticas usa medidas racionais para reduzir o tamanho das páginas, incluindo o uso de compressão, pois os custos da comunicação são mais altos do que os da computação, e maximizando a eficácia do caching. Essa técnica encoraja os sites, especialmente os grandes, a criar versões de seu conteúdo para a Web móvel, pois é apenas isso que é necessário para capturar usuários. Para ajudá-los, há também um logotipo para indicar páginas que podem ser vistas (bem) na Web móvel.

Outra ferramenta útil é uma versão reduzida da HTML, chamada **XHTML Basic**. Essa linguagem é um subconjunto da XHTML, voltado para o uso por telefones móveis, televisores, PDAs, máquinas de venda, pagers, carros, máquinas de jogos e até mesmo relógios. Por esse motivo, ela não admite folhas de estilo, scripts ou frames, mas a maior parte das tags-padrão ainda pode ser usada. Elas são agrupadas em 11 módulos. Alguns são obrigatórios; alguns são opcionais. Todas são definidas em XML. Os módulos e algumas tags de exemplo são listados na Tabela 7.15.

Contudo, nem todas as páginas serão projetadas para funcionar bem na Web móvel. Assim, uma técnica complementar é o uso de **transformação de conteúdo** ou **transcodificação**. Nessa técnica, um computador que se situa entre o dispositivo móvel e o servidor captura as solicitações do dispositivo móvel, busca o conteúdo do servidor e o transforma para o conteúdo Web móvel. Uma transformação simples é reduzir o tamanho de imagens grandes, reformatando-as para uma resolução mais baixa. Muitas outras transformações pequenas, porém úteis, também são possíveis. A transcodificação tem sido usada com algum sucesso desde os primórdios da Web móvel. Veja, por exemplo, Fox et al. (1996). Porém, quando as duas técnicas são usadas, existe uma disputa entre as decisões do conteúdo móvel que são feitas pelo servidor e pelo transcodificador. Por exemplo, um site pode selecionar determinada combinação de imagem e texto para um usuário da Web móvel, mas um transcodificador muda o formato da imagem.

Nossa discussão até aqui tem tratado do conteúdo, e não dos protocolos, pois o conteúdo é o maior problema para a realização da Web móvel. Porém, devemos mencionar rapidamente a questão dos protocolos. Os protocolos HTTP, TCP e IP usados pela Web podem consumir uma largura de banda significativa com overheads de protocolos, como os cabeçalhos. Para resolver esse problema, WAP e outras soluções definiram protocolos de uso especial. Acabou que isso não foi tão necessário. Tecnologias de compactação de cabeçalho, como ROHC (RObust Header Compression), descritas no Capítulo 6, podem reduzir os overheads desses protocolos. Desse modo, é possível ter um conjunto de protocolos (HTTP, TCP, IP) e usá-los por enlaces com largura de banda alta ou baixa. O uso por enlaces com largura de banda baixa apenas requer que a compactação de cabeçalho seja ativada.

7.3.6 Busca na Web

Para encerrar nossa descrição da Web, vamos discutir aquilo que comprovadamente é sua aplicação mais bem-sucedida: a busca. Em 1998, Sergey Brin e Larry Page, na época alunos formados em Stanford, montaram uma empresa chamada Google, para criar um mecanismo melhor de busca na Web. Eles trabalhavam com a ideia então radical de que um algoritmo de busca que contasse quantas vezes cada página era apontada por outras páginas seria uma medida melhor de sua importância do que um que apontasse quantas vezes ela continha as palavras-chave procuradas. Por exemplo, muitas páginas estão vinculadas à página prin-

Módulo	Obrigatório?	Função	Exemplos de tags
Structure	Sim	Estrutura de documento	body, head, html, title
Text	Sim	Informação	br, code, dfn, em, h*n*, kbd, p, strong
Hypertext	Sim	Hiperlinks	a
List	Sim	Listas de itens	dl, dt, dd, ol, ul, li
Forms	Não	Formulários para preenchimento	form, input, label, option, textarea
Tables	Não	Tabelas retangulares	caption, table, td, th, tr
Image	Não	Imagens	img
Object	Não	Applets, mapas etc.	object, param
Meta-information	Não	Informações extras	meta
Link	Não	Semelhante a <a>	link
Base	Não	Ponto de partida do URL	base

Tabela 7.15 | Módulos e tags da XHTML Basic.

cipal da Cisco, o que torna essa página mais importante para um usuário procurando 'Cisco' do que uma página fora da empresa que também use a palavra 'Cisco' muitas vezes.

Eles estavam certos. Foi comprovada a possibilidade de criação de um mecanismo melhor de busca, e as pessoas passaram a utilizá-lo. Com o apoio de capital de risco, a Google cresceu tremendamente. Ela se tornou uma empresa pública em 2004, com uma capitalização de mercado de US$ 23 bilhões. Por volta de 2010, estimava-se que o software fosse executado em mais de um milhão de servidores nos centros de dados do mundo inteiro.

De certa forma, a busca é simplesmente outra aplicação Web, embora uma das mais maduras, pois esteve em desenvolvimento desde os primeiros dias da Web. Porém, a busca provou ser indispensável no uso diário. Estima-se que seja realizado mais de um bilhão de buscas na Web diariamente. Pessoas procurando todo tipo de informação utilizam a busca como ponto de partida. Por exemplo, para descobrir onde comprar Vegemite no Brasil, não existe um site óbvio para usar como ponto de partida. Mas provavelmente um mecanismo de busca conhece uma página com a informação desejada, e pode direcioná-lo rapidamente para a resposta.

Para realizar uma busca na Web pela maneira tradicional, o usuário direciona seu navegador para o URL de um site de busca. Os principais são Google, Yahoo! e Bing. Em seguida, o usuário submete os termos de busca usando um formulário. Esse ato faz com que o mecanismo realize uma consulta em seu banco de dados, procurando páginas ou imagens relevantes, ou qualquer tipo de recurso que esteja sendo procurado, para retornar o resultado como uma página dinâmica. O usuário pode, então, seguir os links para as páginas que foram localizadas.

A busca na Web é um assunto interessante para discussão, pois tem implicações para o projeto e uso de redes. Primeiro, existe a questão de como essa busca encontra páginas. O mecanismo de busca na Web precisa ter um banco de dados de páginas para executar uma consulta. Cada página HTML pode conter links para outras páginas, e tudo que é interessante (ou, pelo menos, pesquisável) está vinculado a algum lugar. Isso significa que, em teoria, é possível começar com algumas páginas e encontrar todas as páginas na Web realizando uma travessia de todas as páginas e links. Esse processo é chamado **Web crawling**. Todos os mecanismos de busca na Web utilizam Web crawlers.

Um problema com o crawling é o tipo de páginas que ele pode encontrar. Buscar documentos estáticos e seguir links é fácil. Porém, muitas páginas Web contêm programas que exibem diferentes páginas, dependendo da interação com o usuário. Um exemplo é um catálogo on-line para um loja. O catálogo pode conter páginas dinâmicas criadas a partir de um banco de dados de produtos e consultas por diferentes produtos. Esse tipo de conteúdo é diferente das páginas estáticas, que são fáceis de atravessar. Como os Web crawlers encontram essas páginas dinâmicas? A resposta é que, na maioria das vezes, eles não encontram. Esse tipo de conteúdo oculto é chamado de **deep Web**. Como pesquisar na deep Web é um problema em aberto, para o qual os pesquisadores agora estão buscando solução. Veja, por exemplo, Madhavan et al. (2008). Também existem convenções pelas quais os sites fazem com que uma página (conhecida como *robots.txt*) diga aos crawlers que partes dos sites devem ou não ser visitadas.

Uma segunda consideração é como processar todos os dados espalhados. Para permitir que algoritmos de indexação sejam executados pela massa de dados, as páginas precisam estar armazenadas. As estimativas variam, mas os principais mecanismos de busca contêm um índice de dezenas de bilhões de páginas retiradas da parte visível da Web. Estima-se que o tamanho médio da página seja de 320 KB. Esses valores significam que uma cópia da Web ocupa algo na ordem de 20 petabytes, ou 2×10^{16} bytes, para ser armazenada. Embora esse seja um número verdadeiramente grande, também é uma quantidade de dados que pode ser armazenada e processada adequadamente nos centros de dados da Internet (Chang et al., 2006). Por exemplo, se o armazenamento em disco custa US$ 20/TB, então 2×10^4 TB custam US$ 400.000, o que não é exatamente um valor muito grande para empresas do tamanho da Google, Microsoft e Yahoo!. E, embora a Web esteja se expandido, os custos de disco estão caindo bastante, de modo que o armazenamento da Web inteira pode continuar a ser viável para grandes empresas em um futuro previsível.

Decifrar esses dados é outra questão. Você pode apreciar como a XML pode ajudar os programas a extrair a estrutura dos dados com facilidade, enquanto os formatos ocasionais exigem muito mais tentativas. Também há o problema de conversão entre formatos, e até mesmo de tradução entre idiomas. Porém, conhecer a estrutura dos dados é apenas parte do problema. O mais difícil é entender o que eles significam. É aí que pode ser encontrado muito valor, começando com páginas de resultados mais relevantes para as consultas. O objetivo principal é poder responder a perguntas, por exemplo, onde comprar um forno elétrico barato, porém decente, na sua cidade.

Um terceiro aspecto da busca na Web é que ela passou a fornecer um nível mais alto de definição de nomes. Não é mais preciso se lembrar de um URL longo se for confiável (ou talvez mais) procurar uma página pelo nome de uma pessoa, supondo que você se lembre melhor de nomes do que de URLs. Essa estratégia é cada vez mais bem-sucedida. Da mesma forma como nomes de DNS relegaram os endereços IP aos computadores, a busca na Web está relegando URLs aos computadores. Também em favor da busca é que ela corrige erros de gramática e digitação, enquanto, se você digitar um URL errado, receberá a página errada.

Finalmente, a busca na Web nos mostra algo que tem pouco a ver com o projeto de redes, mas muito a ver com

o crescimento de alguns serviços da Internet: há muito dinheiro em propaganda. A propaganda é o motor econômico que tem impulsionado o crescimento da busca na Web. A principal mudança da propaganda impressa é a capacidade de direcionar anúncios dependendo do que as pessoas procuram, para aumentar a relevância dos anúncios. São usadas variações em um mecanismo de leilão para que a consulta corresponda aos anúncios mais valiosos (Edelman et al., 2007). Esse novo modelo deu origem a novos problemas, é claro, como a **fraude do clique**, em que os programas imitam os usuários e 'clicam' em anúncios, para causar pagamentos que não foram ganhos honestamente.

7.4 Streaming de áudio e vídeo

As aplicações da Web e a Web móvel não são os únicos desenvolvimentos interessantes no uso de redes. Para muitas pessoas, áudio e vídeo são o Santo Graal das redes. Quando a palavra 'multimídia' é mencionada, tanto os mais talentosos quanto os empresários começam a ficar com água na boca. O primeiro grupo consegue ver imensos desafios técnicos ao oferecer voz sobre IP e vídeo por demanda a cada computador. O últimos enxergam enormes lucros com isso.

Embora a ideia de enviar áudio e vídeo pela Internet já exista pelo menos desde a década de 70, somente por volta de 2000 é que o tráfego de **áudio em tempo real** e de **vídeo em tempo real** cresceu mais intensivamente. O tráfego em tempo real é diferente do tráfego da Web porque precisa ser reproduzido em alguma velocidade predeterminada para que seja útil. Afinal, assistir a um vídeo em câmera lenta, com várias interrupções, não é uma ideia de diversão para a maioria das pessoas. Ao contrário, a Web pode ter interrupções curtas, e os carregamentos de página podem levar mais ou menos tempo, dentro de certos limites, sem que isso cause um grande problema.

Duas coisas aconteceram para permitir esse crescimento. Primeiro, os computadores se tornaram muito mais poderosos e são equipados com microfones e câmeras, de modo que podem inserir, processar e enviar dados de áudio e vídeo com facilidade. Segundo, uma enxurrada de largura de banda da Internet passou a estar disponível. Enlaces de longa distância no núcleo da Internet utilizam muitos gigabits/s, e as redes de banda larga e sem fios 802.11 alcançam usuários na borda da Internet. Esses desenvolvimentos permitem que os ISPs executem grandes níveis de tráfego por seus backbones e significam que os usuários comuns podem se conectar à Internet de 100 a 1.000 vezes mais rápido do que com um modem de telefone a 56 kbps.

O aumento de largura de banda causou o crescimento do tráfego de áudio e vídeo, mas por motivos diferentes. As ligações telefônicas ocupam relativamente pouca largura de banda (em princípio, 64 kbps, porém menos quando compactadas), embora o serviço telefônico tradicionalmente seja caro. As empresas viram uma oportunidade para transportar o tráfego de voz pela Internet usando a largura de banda existente, para reduzir suas contas telefônicas. Empresas como Skype viram um modo de permitir que os clientes façam ligações telefônicas gratuitas usando suas conexões com a Internet. Companhias telefônicas que surgiram do nada viram um modo barato de realizar chamadas de voz tradicionais usando equipamento de rede IP. O resultado foi uma explosão de dados de voz transportados pelas redes da Internet, o que é chamado **Voice over IP** ou **telefonia via Internet**.

Diferentemente do áudio, o vídeo ocupa uma grande quantidade de largura de banda. O vídeo pela Internet com qualidade razoável é codificado com compressão em taxas de aproximadamente 1 Mbps, e um filme de DVD típico contém 2 GB de dados. Antes do acesso à Internet por banda larga, o envio de filmes pela rede era proibitivo. Isso não é mais assim. Com a ampla utilização da banda larga, foi possível pela primeira vez que os usuários assistissem em casa a um vídeo por demanda com qualidade decente. As pessoas gostam muito disso. Estima-se que cerca de um quarto dos usuários da Internet visite diariamente o YouTube, um site popular de compartilhamento de vídeo. O negócio de aluguel de filmes tem se deslocado para downloads on-line. E o grande tamanho dos vídeos tem mudado a composição geral do tráfego da Internet. A maior parte do tráfego da Internet já é de vídeo, e estima-se que 90 por cento do tráfego da Internet será também de vídeo dentro de alguns anos (Cisco, 2010).

Como existe largura de banda suficiente para transportar áudio e vídeo, a questão principal para desenvolver aplicações de streaming de mídia e teleconferência é o atraso de rede. Áudio e vídeo precisam de apresentação em tempo real, significando que eles têm de ser reproduzidos em uma velocidade predeterminada para que sejam úteis. Longos atrasos significam que as chamadas que deveriam ser interativas deixam de sê-lo. Esse problema é claro se você já tiver falado em um telefone por satélite, quando o atraso de subida de meio segundo é bastante incômodo. Para tocar música e filmes pela rede, o atraso absoluto não importa, pois afeta apenas quando a mídia começa a ser reproduzida. Mas a variação no atraso, chamada **jitter**, ainda importa. Ela precisa ser mascarada pelo player ou então o áudio não será inteligível e o vídeo terá interrupções.

Nesta seção, discutiremos algumas estratégias para lidar com o problema do atraso, bem como os protocolos para estabelecer sessões de áudio e vídeo. Após uma introdução ao áudio e vídeo digitais, nossa apresentação se divide em três casos para os quais são usados diferentes projetos. O primeiro caso, e mais fácil de tratar, é o streaming de mídia armazenada, como assistir a um vídeo pelo YouTube. O próximo caso em termos de dificuldade é o streaming de mídia ao vivo. Dois exemplos são rádio via Internet e IPTV, em que estações de rádio e televisão transmitem para muitos usuários ao vivo na Internet. O

último caso, e mais difícil, é uma chamada feita com o Skype ou, de forma mais generalizada, uma conferência interativa de áudio e vídeo.

A propósito, o termo **multimídia** normalmente é usado no contexto da Internet para indicar vídeo e áudio. Literalmente, multimídia significa dois ou mais tipos de mídia. Essa definição torna este livro uma apresentação em multimídia, pois contém texto e gráficos (as figuras). Porém, provavelmente não era isso que você tinha em mente, e portanto usamos o termo 'multimídia' para indicar dois ou mais tipos de **mídia contínua**, ou seja, que precisa ser reproduzida durante algum intervalo de tempo bem definido. Os dois tipos de mídia normalmente são vídeo com áudio, isto é, imagens em movimento com som. Muitas pessoas se referem ao áudio puro, como a telefonia via Internet ou o rádio via Internet, também como multimídia, o que logicamente não é exato. Na realidade, um termo melhor para todos esses casos é **streaming de mídia**. Apesar disso, vamos acompanhar a maioria e considerar o áudio em tempo real como sendo também multimídia.

7.4.1 Áudio digital

Um sinal de áudio (som) é uma onda acústica unidimensional (de pressão). Quando uma onda acústica entra no ouvido, o tímpano vibra, fazendo com que os minúsculos ossos do ouvido interno vibrem também, enviando impulsos nervosos ao cérebro. Esses pulsos são percebidos como sons pelo ouvinte. Da mesma forma, quando uma onda acústica chega a um microfone, ele produz um sinal elétrico, representando a amplitude do som como uma função do tempo.

A faixa de frequências sonoras captadas pelo ouvido humano começa em 20 e chega a 20.000 Hz. Alguns animais, em especial os cães, podem ouvir frequências mais altas. Os ouvidos percebem os sons no modo logarítmico; portanto, a razão de dois sons com potência *A* e *B* é, por convenção, expressa em **dB (decibéis)**, como a quantidade $10 \log_{10}(A/B)$. Se definirmos o limite mínimo de audibilidade (uma pressão sonora de cerca de 20 μpascals) para uma onda senoidal de 1 KHz como 0 dB, uma conversa normal terá cerca de 50 dB, e o limite máximo tolerável será de aproximadamente 120 dB, um intervalo dinâmico da ordem de um milhão.

O ouvido humano é surpreendentemente sensível a variações de som que duram apenas milésimos de segundo. O olho, ao contrário, não percebe mudanças no nível da luz que durem por menos de alguns milissegundos. O resultado dessa observação é que a flutuação de apenas alguns milésimos de segundo durante uma transmissão de multimídia afeta mais a qualidade do som percebido do que a qualidade da imagem percebida.

O áudio digital é uma representação digital de uma onda de áudio que pode ser usada para recriá-lo. As ondas de áudio podem ser convertidas para a forma digital por um conversor analógico-digital, ou **ADC (Analog-to-Digital Converter)**. Um ADC recebe uma tensão elétrica como entrada e gera um número binário como saída. Na Figura 7.16(a) podemos ver um exemplo de uma onda senoidal. Para representar esse sinal na forma digital, podemos obter amostras a cada *T* segundos, como indicam as barras da Figura 7.16(b). Se um sinal sonoro não for uma onda senoidal pura, mas sim uma superposição linear de ondas senoidais, em que o componente de frequência mais alta presente é *f*, o teorema de Nyquist (ver Capítulo 2) diz que é suficiente obter amostras a uma frequência 2*f*. Usar uma taxa de amostragem maior não é interessante, pois as frequências mais altas que essa amostragem poderia detectar não estão presentes.

O processo inverso captura valores digitais e produz uma tensão elétrica analógica. Isso é feito por um conversor digital-analógico, ou **DAC** (**Digital-to-Analog Converter**). Um alto-falante pode, então, converter a tensão analógica em ondas acústicas, para que as pessoas possam ouvir os sons.

As amostras digitais nunca são exatas. As amostras da Figura 7.16(c) permitem apenas nove valores, de −1,00 até 1,00 em intervalos de 0,25. Uma amostra de 8 bits permitiria 256 valores distintos. Uma amostra de 16 bits permitiria 65.536 valores distintos. O erro introduzido pelo número finito de bits por amostra é chamado **ruído de quantização**. Se ele for muito grande, o ouvido poderá detectá-lo.

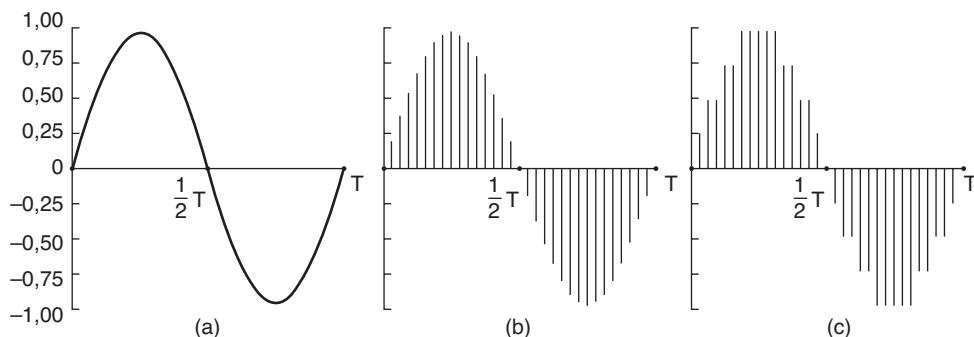



Figura 7.16 | (a) Uma onda senoidal. (b) Amostragem da onda senoidal. (c) Quantização das amostras para 4 bits.

Dois exemplos bem conhecidos em que a amostragem do som é usada são os telefones e os CDs de áudio. A modulação por código de pulso, como era usada no sistema telefônico, utiliza amostras de 8 bits geradas 8 mil vezes por segundo. A escala é não linear para minimizar a distorção percebida, e, com apenas 8 mil amostras/s, as frequências acima de 4 KHz são perdidas. Na América do Norte e no Japão, a codificação utilizada é **μ-law**. Na Europa e internacionalmente, a codificação utilizada é **A-law**. Cada codificação gera uma taxa de dados de 64.000 bps.

Os CDs convencionais (de áudio) são digitais com uma taxa de amostragem de 44.100 amostras por segundo, o suficiente para capturar frequências de até 22.050 Hz, o que é bom para os seres humanos mas ruim para os cães. As amostras têm 16 bits cada uma e são lineares na faixa de amplitudes. Observe que as amostras de 16 bits só permitem 65.536 valores distintos, embora a amplitude dinâmica do ouvido seja de cerca de 1 milhão. Assim, embora o áudio com qualidade de CD seja muito melhor do que o áudio com qualidade de telefone, o uso de 16 bits por amostra introduz algum ruído de quantização (ainda que a amplitude dinâmica total não seja utilizada — os CDs não foram feitos para prejudicar a audição das pessoas.) Alguns audiófilos fanáticos ainda preferem discos de LP a 33 RPM aos CDs, pois os discos não possuem um corte de frequência de Nyquist a 22 KHz e não possuem ruído de quantização. (Mas eles têm arranhões, a menos que sejam manipulados com muito cuidado.) Com 44.100 amostras por segundo de 16 bits cada uma, o áudio não comprimido com qualidade de CD precisa de uma largura de banda de 705,6 kbps para som monofônico e 1,411 Mbps para som estéreo.

Compressão de áudio

O áudio normalmente é comprimido para reduzir as necessidades de largura de banda e tempos de transmissão, embora as taxas de dados de áudio sejam muito menores do que as de dados de vídeo. Todos os sistemas de compressão exigem dois algoritmos: um para comprimir os dados na origem e outro para descomprimi-los no destino. Na literatura, esses algoritmos são conhecidos como algoritmos de **codificação** e **decodificação**, respectivamente. Usaremos essa terminologia também.

Os algoritmos de compressão exibem certas assimetrias importantes de ser entendidas. Embora estejamos considerando primeiro o áudio, essas assimetrias se mantêm também para o vídeo. Para muitas aplicações, um documento de multimídia só será codificado uma vez (quando for armazenado no servidor de multimídia), mas será decodificado milhares de vezes (quando for reproduzido pelos clientes). Essa assimetria significa que é aceitável que o algoritmo de codificação seja lento e exija um hardware caro, desde que o algoritmo de decodificação seja rápido e não exija hardware caro. O operador de um servidor de áudio (ou vídeo) popular poderia estar disposto a comprar um cluster de computadores para codificar sua biblioteca inteira, mas, se exigir que os clientes façam o mesmo para escutar a música ou assistir aos filmes, provavelmente não terá muito sucesso. Muitos sistemas de compressão práticos se esforçam bastante para tornar a decodificação rápida e simples, mesmo ao preço de tornar a codificação lenta e complicada.

Por outro lado, para o áudio e o vídeo ao vivo, como em chamadas de VoIP, a codificação lenta é inaceitável. A codificação precisa acontecer na mesma hora, em tempo real. Consequentemente, multimídia em tempo real usa diferentes algoritmos ou parâmetros que o áudio ou vídeo armazenados em disco, normalmente com bem menos compressão.

Uma segunda assimetria é que o processo de codificação/decodificação não precisa ser reversível. Ou seja, ao comprimir um arquivo de dados, transmiti-lo e depois descomprimi-lo, o usuário espera receber de volta o original, exatamente até o último bit. Com a multimídia, esse requisito não existe. Em geral, é aceitável ter um sinal de áudio (ou vídeo) após a codificação e depois decodificá-lo de forma ligeiramente diferente do original, desde que o som ou a imagem sejam os mesmos. Quando a saída decodificada não é exatamente igual à entrada original, o sistema é considerado **com perdas**. Se a entrada e saída forem idênticas, o sistema é **sem perdas**. Os sistemas com perdas são importantes porque aceitar uma pequena quantidade de perda de informação normalmente significa um grande retorno em termos de taxa de compressão possível.

Historicamente, a largura de banda a longa distância na rede telefônica era muito cara, de modo que existe um trabalho substancial em **vocoders** (abreviação de 'voice coders', ou codificadores de voz), que comprimem o áudio para o caso especial da fala. A fala humana tende a estar na faixa de 600 a 6.000 Hz, e é produzida por um processo mecânico que depende do trato vocal, da língua e do maxilar de quem fala. Alguns vocoders utilizam modelos do sistema vocal para reduzir a fala a alguns parâmetros (por exemplo, os tamanhos e formas das diversas cavidades) e uma taxa de dados tão pequena quanto 2,4 kbps. Contudo, o modo de funcionalidade desses vocoders está fora do escopo deste livro.

Vamos nos concentrar no áudio enviado pela Internet, que costuma ser mais próximo da qualidade de CD. Também é desejável reduzir as taxas de dados para esse tipo de áudio. A 1,411 Mbps, o áudio estéreo amarraria muitos enlaces de banda larga, deixando menos espaço para vídeo e outros tipos de tráfego da Web. Sua taxa de dados com compressão pode ser reduzida por uma ordem de grandeza com pouca ou nenhuma perda de qualidade perceptível.

A compressão e a descompressão exigem processamento de sinal. Felizmente, o som e os filmes digitalizados podem ser facilmente processados pelos softwares nos computadores. De fato, existem dezenas de programas para permitir que os usuários gravem, exibam, editem, misturem

e armazenem mídia de várias origens. Isso tem ocasionado grandes quantidades de música e filmes disponíveis na Internet — nem todos legais —, o que resultou em diversos processos de artistas e proprietários de direito autoral.

Muitos algoritmos de compressão de áudio foram desenvolvidos. Provavelmente, os formatos mais populares são **MP3** (**MPEG audio layer 3**) e **AAC** (**Advanced Audio Coding**), transportado em arquivos **MP4 (MPEG-4)**. Para evitar confusão, observe que MPEG oferece compressão de áudio e vídeo. MP3 refere-se à parte de compressão de áudio (parte 3) do padrão MPEG-1, e não à terceira versão do MPEG. De fato, nenhuma terceira versão do MEPG foi lançada, apenas MPEG-1, MPEG-2 e MPEG-4. AAC é o sucessor para MP3 e a codificação de áudio padrão usada em MPEG-4. MPEG-2 permite áudio MP3 e AAC. Ficou claro agora? A melhor coisa a respeito de padrões é que existem muitos para escolher. E, se você não gostar de nenhum deles, basta esperar um ou dois anos.

A compressão de áudio pode ser feita de duas maneiras. Na **codificação da forma de onda** o sinal é modificado matematicamente por uma transformação de Fourier em seus componentes de frequência. No Capítulo 2, mostramos um exemplo de função no tempo e suas amplitudes de Fourier na Figura 2.1(a). A amplitude de cada componente é então codificada de modo mínimo. O objetivo é reproduzir com precisão a forma de onda na outra extremidade, com a menor quantidade de bits possível.

O outro modo, chamado **codificação perceptiva**, explora certas falhas no sistema auditivo para codificar um sinal que soe da mesma forma para um ouvinte humano, ainda que pareça bem diferente em um osciloscópio. A codificação perceptiva se baseia na ciência da **psicoacústica** — a forma como as pessoas percebem o som. MP3 e AAC são baseados na codificação perceptiva.

A principal propriedade da codificação perceptiva é que alguns sons podem **mascarar** outros. Imagine que você esteja transmitindo um concerto de flauta ao vivo em um dia quente de verão. De repente, um equipe de trabalhadores liga suas britadeiras e começa a esburacar a rua. Ninguém mais consegue ouvir a flauta. Seu som foi mascarado pelas britadeiras. Para fins de transmissão, agora é suficiente codificar apenas a banda de frequência usada pelas britadeiras, porque os ouvintes não podem mais ouvir a flauta. Isso se chama **máscara de frequência** — a habilidade de um som em alto volume em uma banda de frequência ocultar um som mais suave em outra banda de frequência que seria audível na ausência do som alto. De fato, mesmo depois que as britadeiras pararem, a flauta será inaudível por um curto período de tempo, porque o ouvido diminuiu seu ganho enquanto elas funcionavam e demora um tempo finito para aumentar novamente o ganho. Esse efeito é chamado **máscara temporal**.

Para tornar esses efeitos mais quantitativos, imagine a experiência 1. Uma pessoa em um quarto silencioso coloca fones de ouvido conectados à placa de som de um computador. O computador gera uma onda senoidal pura a 100 Hz em potência baixa, mas gradualmente crescente. A pessoa é instruída a pressionar uma tecla ao ouvir o tom. O computador registra o nível de potência atual e depois repete a experiência a 200 Hz, 300 Hz e em todas as outras frequências, até o limite da audição humana. Quando é calculada a média para várias pessoas, o resultado é um gráfico logarítmico que mostra a potência necessária para que um tom seja audível, semelhante ao gráfico da Figura 7.17(a). Uma consequência direta dessa curva é que nunca se torna necessário codificar quaisquer frequências cuja potência fique abaixo do limiar de audibilidade. Por exemplo, se a potência a 100 Hz fosse 20 dB na Figura 7.17(a), ela poderia ser omitida da saída sem nenhuma perda perceptível de qualidade, porque 20 dB a 100 Hz fica abaixo do nível de audibilidade.

Agora considere o experimento 2. O computador executa o experimento 1 novamente, mas desta vez com uma onda senoidal de amplitude constante, digamos, 150 Hz, sobreposta na frequência de teste. O que descobrimos é que o limiar de audibilidade para frequências próximas de 150 Hz aumenta, como mostra a Figura 7.17(b).

A consequência dessa nova observação é que, registrando quais sinais estão sendo mascarados por sinais mais poderosos nas bandas de frequência nas proximidades, podemos omitir mais e mais frequências no sinal codificado, economizando bits. Na Figura 7.17, o sinal de 125 Hz pode

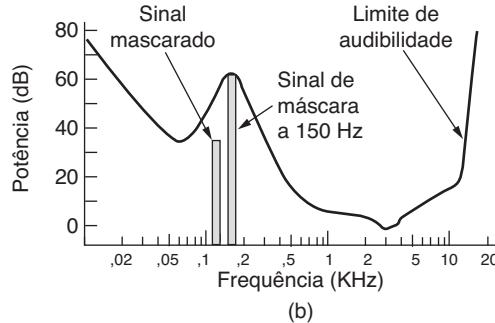
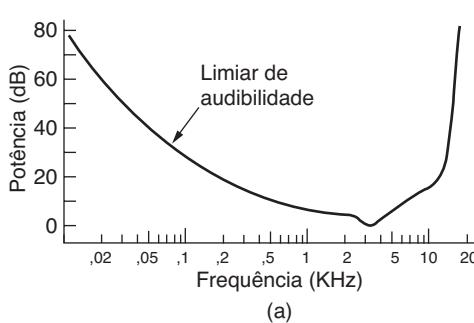



Figura 7.17 | (a) O limiar de audibilidade como função da frequência. (b) O efeito de máscara.

ser completamente omitido da saída e ninguém poderá ouvir a diferença. Mesmo depois que um sinal poderoso termina em alguma banda de frequência, o conhecimento de suas propriedades de máscara temporal nos permite continuar a omitir as frequências mascaradas por algum intervalo de tempo, enquanto o ouvido se recupera. A essência do MP3 e AAC é a transformação de Fourier do som para obter a potência em cada frequência e depois transmitir apenas as não mascaradas, codificando-as com o mínimo de bits possível.

Com essas informações, agora podemos ver como a codificação é feita. A compressão de áudio é realizada fazendo-se a amostragem da forma de onda de 8 a 96 KHz para AAC, normalmente a 44,1 KHz, para imitar o som do CD. A amostragem pode ser feita em um (mono) ou dois canais (estéreo). Em seguida, a taxa de bits é escolhida. O MP3 pode compactar um CD de rock até 96 kbps com pouca perda de qualidade perceptível, mesmo para os fãs de rock sem nenhuma deficiência auditiva. Para um concerto de piano, são necessários pelo menos 128 kbps com AAC. A diferença ocorre porque a relação sinal/ruído do rock é muito mais alta que a de um concerto de piano (no sentido da engenharia). Também é possível escolher taxas de saída mais baixas e aceitar uma certa perda de qualidade.

Em seguida, as amostras são processadas em grupos. Cada grupo passa por um banco de filtros digitais para gerar bandas de frequência. A informação de frequência é entregue a um modelo psicoacústico para se determinarem as frequências mascaradas. Depois, a quantidade de bits disponível é dividida entre as bandas, com mais bits alocados às bandas com a maior potência espectral não mascarada, menos bits alocados a bandas não mascaradas com menor potência espectral e nenhum bit alocado a bandas mascaradas. Finalmente, os bits são codificados com o emprego da codificação de Huffman, que atribui códigos curtos a números que aparecem com frequência e códigos longos aos que ocorrem com pouca frequência. Na realidade, há outros detalhes. Para obter mais informações, consulte Brandenburg (1999).

7.4.2 Vídeo digital

Agora que sabemos tudo sobre o ouvido humano, é hora de prosseguirmos para o olho. (Só para informar: não existe uma próxima seção sobre o nariz.) O olho humano tem a propriedade de, quando uma imagem aparece na retina, retê-la por alguns milissegundos antes que desapareça. Se uma sequência de imagens for apresentada a 50 imagens/s, o olho não nota que está vendo imagens diferentes. Todos os sistemas de vídeo exploram esse princípio para produzir imagens em movimento.

A representação digital mais simples do vídeo é uma sequência de quadros, cada um consistindo em uma grade retangular de elementos de imagem, ou **pixels**. Cada pixel pode ser um único bit, para representar ou preto ou branco. Porém, a qualidade desse sistema é terrível. Tente usar seu editor de imagens favorito para converter os pixels de uma imagem colorida para preto e branco (e *não* tonalidades de cinza).

O próximo passo é usar 8 bits por pixel para representar 256 níveis de cinza. Esse esquema oferece vídeo 'preto e branco' de maior qualidade. Para o vídeo colorido, muitos sistemas usando 8 bits para cada um dos componentes de cor primária — vermelho, verde e azul (RGB). Essa representação é possível porque qualquer cor pode ser construída com base em uma sobreposição linear de vermelho, verde e azul com as intensidades apropriadas. Com 24 bits por pixel, há cerca de 16 milhões de cores possíveis, o que é mais do que o olho humano consegue distinguir.

Em monitores de computador LCD coloridos e em aparelhos de televisão, cada pixel distinto é composto de subpixels vermelho, verde e azul muito próximos. Os quadros são exibidos definindo-se a intensidade dos subpixels, e o olho humano mistura os componentes da cor.

Taxas de quadros comuns são 24 quadros/s (herdada do filme de cinema de 35 mm), 30 quadros/s (herdada do sistema de televisão NTSC dos Estados Unidos) e 25 quadros/s (herdada do sistema de televisão PAL, usado em quase todo o restante do mundo). (Para os mais detalhistas, o sistema de televisão em cores NTSC usa 29,97 quadros/s. O sistema original em preto e branco usava 30 quadros/s, mas, quando a cor foi introduzida, os engenheiros precisaram de um pouco mais de largura de banda para a sinalização, e por isso reduziram a taxa de quadros para 29,97. Os vídeos NTSC para computadores realmente utilizam 30. O sistema PAL foi inventado após NTSC e, na realidade, usa 25,00 quadros/s. Para completar essa história, um terceiro sistema, SECAM, é usado na França, em países da África de língua francesa e na Europa oriental. Ele foi introduzido na Europa oriental pela então Alemanha Oriental (República Democrática Alemã, comunista), para que seus cidadãos não pudessem assistir à televisão da Alemanha Ocidental (República Federal Alemã, não comunista) (PAL), por receio de que tivessem más ideias. Mas muitos desses países estão passando para PAL. Tecnologia e política no melhor estilo.

Na realidade, para a televisão por radiodifusão, 25 quadros/s não é bom o suficiente para um movimento suave, de modo que as imagens são divididas em dois **campos**, um com as linhas de varreduras de número ímpar e outro com as linhas de varredura pares. Os dois campos (meia resolução) são enviados sequencialmente, gerando quase 60 (NTSC) ou exatamente 50 (PAL) campos/s, um sistema conhecido como **entrelaçamento**. Os vídeos para exibição em um computador são **progressivos**, ou seja, eles não usam o entrelaçamento, pois os monitores de computador possuem buffers em suas placas gráficas, possibilitando que a CPU coloque uma nova imagem no buffer 30 vezes/s, mas a placa gráfica redesenha a tela a cada 50 ou até mesmo 100 vezes/s, para eliminar a cintilação. Os aparelhos de

televisão analógicos não possuem um buffer de quadros, assim como os computadores. Quando um vídeo entrelaçado com movimento rápido é exibido em um computador, linhas horizontais curtas são visíveis perto das arestas, um efeito conhecido como **combinação**.

Os tamanhos de quadro usados para o vídeo enviado pela Internet variam muito pelo simples motivo de que quadros maiores exigem mais largura de banda, que nem sempre pode estar disponível. O vídeo com resolução baixa pode ser de 320 por 240 pixels, e o vídeo de 'tela cheia' é de 640 por 480 pixels. Essas dimensões são próximas daquelas usadas nos antigos monitores de computador e televisão NTSC, respectivamente. A **relação entre eixos** (ou relação entre largura e altura) de 4:3 é a mesma que a usada na televisão-padrão. Vídeos de **HDTV** (**High-Definition TeleVision**) podem ser baixados com 1.280 por 720 pixels. Essas imagens de 'tela larga' possuem uma relação entre eixos de 16:9, para combinar melhor com a relação 3:2 do filme. Por comparação, os vídeos em DVD-padrão normalmente usam 720 por 480 pixels, e o vídeo em discos Blu-ray normalmente é HDTV, a 1.080 por 720 pixels.

Na Internet, o número de pixels é apenas parte da história, pois os players de mídia podem apresentar a mesma imagem em diferentes tamanhos. O vídeo é apenas outra janela em uma tela de computador, que pode ser aumentada ou encolhida. O papel de mais pixels é aumentar a qualidade da imagem, de modo que não apareça borrada quando expandida. Porém, muitos monitores podem mostrar imagens (e, portanto, vídeos) com até mesmo mais pixels do que HDTV.

Compressão de vídeo

Já deve estar claro, pela nossa discussão sobre vídeo digital, que a compressão é fundamental para o envio de vídeo pela Internet. Até mesmo o vídeo com quadros de qualidade-padrão de 640 por 480 pixels, 24 bits de informações de cor por pixel e 30 quadros/s exige mais de 200 Mbps. Isso é muito superior à largura de banda pela qual a maioria dos escritórios de empresa está conectada à Internet, quanto mais usuários domésticos, e tudo isso para um único streaming de vídeo. Como a transmissão de vídeo não compactado está absolutamente fora de questão, pelo menos em redes a longas distâncias, a única esperança é que uma compressão maciça seja possível. Felizmente, as pesquisas nas últimas décadas nos levaram a muitas técnicas e algoritmos de compressão que tornam a transmissão de vídeo viável.

Muitos formatos são usados para o vídeo que é enviado pela Internet, alguns fechados e outros padronizados. A codificação mais popular é MPEG em suas várias formas. Ele é um padrão aberto, encontrado em arquivos com extensões mpg e mp4, além de outros formatos com suporte ao MPEG. Nesta seção, examinaremos MPEG para estudar como a compressão de vídeo é realizada. Para começar, veremos a compressão de imagens paradas com JPEG. Um vídeo é simplesmente uma sequência de imagens (mais som). Um modo de comprimir o vídeo é codificar cada imagem em sequência. Até certo ponto, MPEG é apenas a codificação JPEG de cada quadro, mais alguns recursos extras para remover a redundância entre os quadros.

O padrão JPEG

O padrão **JPEG** (**Joint Photographic Experts Group**) para compressão de imagens paradas em tom contínuo (por exemplo, fotografias) foi desenvolvido por especialistas fotográficos trabalhando sob os auspícios conjuntos da ITU, ISO e IEC, outra agência de padrões. Ele é bastante usado (procure arquivos com a extensão jpg) e normalmente oferece razões de compressão de 10:1 ou ainda mais para imagens naturais.

JPEG é definido no International Standard 10918. Na realidade, ele é mais uma lista de compras do que um único algoritmo, porém, dos quatro modos que são definidos, apenas o modo sequência com perdas é relevante para nossa discussão. Além do mais, vamos nos concentrar no modo como o JPEG normalmente é usado para codificar imagens de vídeo RGB de 24 bits e omitiremos algumas das opções e detalhes para simplificar.

O algoritmo é ilustrado na Figura 7.18. A etapa 1 é a preparação do bloco. Por uma questão de especificidade, vamos supor que a entrada JPEG seja uma imagem RGB de 640 × 480 com 24 bits/pixel, como mostra a Figura 7.19(a). RGB não é o melhor modelo de cor para usar na compressão. O olho humano é muito mais sensível à **luminescência**, ou brilho, dos sinais de vídeo do que a **crominância**, ou cor dos sinais de vídeo. Assim, primeiro calculamos a luminescência, Y, e as duas crominâncias, Cb e Cr, a partir dos componentes R, G e B. As fórmulas a seguir são usadas para valores de 8 bits que variam de 0 a 255:

$$Y = 16 + 0{,}26R + 0{,}50G + 0{,}09B$$

$$Cb = 128 + 0{,}15R - 0{,}29G - 0{,}44B$$

$$Cr = 128 + 0{,}44R - 0{,}37G + 0{,}07B$$

Figura 7.18 | A operação do JPEG em modo sequencial com perdas.

São criadas matrizes separadas para *Y*, *Cb* e *Cr*. Em seguida, são usados blocos quadrados de quatro pixels para a divisão proporcional das matrizes de *Cb* e *Cr*, a fim de reduzi-las a 320 × 240. Há perdas nessa redução, que são praticamente imperceptíveis, pois o olho humano reage mais à luminescência que à crominância. Contudo, essa redução compacta os dados em um fator de dois. Portanto, 128 é subtraído de cada elemento das três matrizes para inserir 0 no meio da faixa. Por último, cada matriz é dividida em blocos de 8 × 8. A matriz *Y* tem 4.800 blocos; as outras duas têm 1.200 blocos cada uma, como mostra a Figura 7.19(b).

O segundo passo da codificação JPEG é aplicar uma transformação cosseno discreta, ou **DCT** (**Discrete Cosine Transformation**), a cada um dos 7.200 blocos separadamente. A saída de cada DCT é uma matriz de 8 × 8 de coeficientes DCT. O elemento DCT (0,0) é o valor médio do bloco. Os outros elementos informam quanta potência espectral está presente em cada frequência espacial. Normalmente, esses elementos desaparecem com rapidez com o aumento da distância em relação à origem (0,0), como sugere a Figura 7.20.

Depois que a DCT se completa, o JPEG passa para a etapa 3, chamada **quantização**, na qual os coeficientes DCT menos importantes são descartados. Essa transformação (com perdas) é realizada por meio da divisão de cada coeficiente da matriz DCT 8 × 8 por um peso obtido em uma tabela. Se todos os pesos forem iguais a 1, a transformação nada fará. Entretanto, se os pesos aumentarem muito a partir da origem, frequências espaciais mais altas serão abandonadas rapidamente.

Um exemplo dessa etapa é mostrado na Figura 7.21. Nela, vemos a matriz DCT inicial, a tabela de quantização e o resultado obtido ao dividir cada elemento DCT pelo elemento correspondente na tabela de quantização. Os valores na tabela de quantização não fazem parte do padrão JPEG. Cada aplicação deve fornecer seus próprios valores, garantindo assim o controle da negociação entre perdas e compressão.

O passo 4 reduz o valor (0,0) de cada bloco (o valor do canto superior esquerdo) substituindo-o pela diferença entre ele e o elemento correspondente no bloco anterior. Como são as médias de seus respectivos blocos, esses elementos devem mudar lentamente; portanto, ao retirarmos os valores diferenciais, devemos reduzir a maioria desses elementos a valores pequenos. Nenhum diferencial é calculado a partir dos outros valores.

Em seguida, a etapa 5 torna lineares os 64 elementos e aplica a codificação run-length à lista. A varredura do bloco da esquerda para a direita e depois de cima para baixo não fará com que os zeros sejam concentrados e, portanto, é usado um padrão de varredura em ziguezague, como mostra a Figura 7.22. Nesse exemplo, o padrão em ziguezague

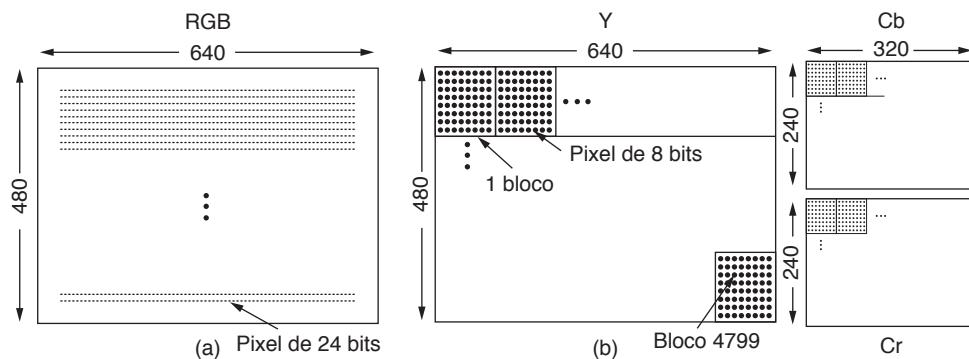



Figura 7.19 | (a) Dados RGB de entrada. (b) Depois da preparação do bloco.

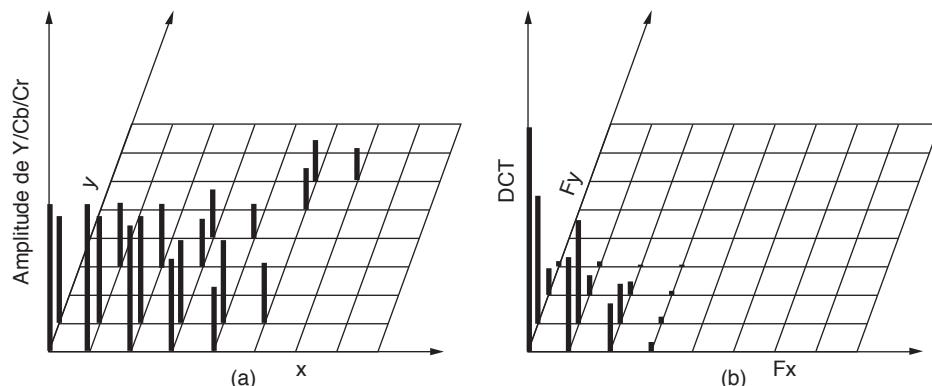



Figura 7.20 | Um bloco da matriz Y. (b) Os coeficientes DCT.

Coeficientes DCT

150	80	40	14	4	2	1	0
92	75	36	10	6	1	0	0
52	38	26	8	7	4	0	0
12	8	6	4	2	1	0	0
4	3	2	0	0	0	0	0
2	2	1	1	0	0	0	0
1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Tabela de quantização

1	1	2	4	8	16	32	64
1	1	2	4	8	16	32	64
2	2	2	4	8	16	32	64
4	4	4	4	8	16	32	64
8	8	8	8	8	16	32	64
16	16	16	16	16	16	32	64
32	32	32	32	32	32	32	64
64	64	64	64	64	64	64	64

Coeficientes quantizados

150	80	20	4	1	0	0	0
92	75	18	3	1	0	0	0
26	19	13	2	1	0	0	0
3	2	2	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Figura 7.21 | Cálculo dos coeficientes DCT quantizados.

acaba produzindo 38 zeros consecutivos no fim da matriz. Essa sequência pode ser reduzida a uma simples contagem informando que há 38 zeros, uma técnica conhecida como **codificação run-length** (tamanho corrido).

Agora, temos uma lista de números que representam a imagem (em espaço de transformação). A etapa 6 é a codificação de Huffman dos números para armazenamento ou transmissão, atribuindo códigos mais curtos aos números comuns e mais longos aos números pouco comuns.

O JPEG talvez pareça complicado, mas isso acontece porque de fato ele *é* complicado. Mesmo assim, a compressão de 20:1 ou mais ainda compensa. A decodificação de uma imagem JPEG exige a execução do algoritmo invertido. O JPEG é aproximadamente simétrico: a decodificação demora tanto quanto a codificação. Essa propriedade não é verdadeira para todos os algoritmos de compressão, como veremos agora.

O padrão MPEG

Finalmente, chegamos ao centro da questão: os padrões **MPEG (Motion Picture Experts Group).** Embora existam muitos algoritmos fechados, esses padrões definem os principais algoritmos usados para a compressão de vídeos e têm sido adotados como padrões internacionais desde 1993. Como os filmes contêm tanto sons quanto imagens,

150	80	20	4	1	0	0	0
92	75	18	3	1	0	0	0
26	19	13	2	1	0	0	0
3	2	2	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

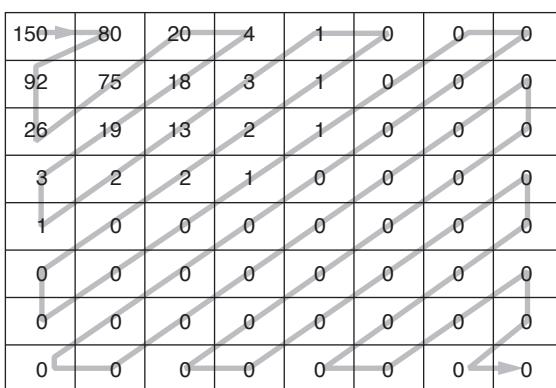
Figura 7.22 | A ordem na qual os valores quantizados são transmitidos.

o MPEG pode compactar igualmente áudio e vídeo. Já examinamos a compressão de áudio e a compressão de imagens estáticas; portanto, vamos estudar agora a compressão de vídeo.

O padrão MPEG-1 (que inclui áudio MP3) foi publicado inicialmente em 1993 e ainda é bastante utilizado. Seu objetivo era produzir uma saída com qualidade de gravador de vídeo que fosse comprimida à razão de 40:1 para taxas em torno de 1 Mbps. Esse vídeo é adequado para uso na Internet em geral nos sites Web. Não se preocupe se você não se lembra dos gravadores de vídeo — MPEG-1 também foi usado para armazenar filmes nos CDs, quando eles existiam. Se você não sabe o que são CDs, temos que prosseguir para MPEG-2.

Lançado em 1996, o padrão MPEG-2 foi projetado para compactar vídeos com qualidade de broadcast. Ele é muito comum agora, pois é usado como base para o vídeo codificado em DVDs (que inevitavelmente tem lugar na Internet) e para a televisão por broadcast digital (como DVB). O vídeo com qualidade de DVD em geral é codificado em taxas de 4 a 8 Mbps.

O padrão MPEG-4 tem dois formatos de vídeo. O primeiro, lançado em 1999, codifica vídeo com uma representação baseada em objeto. Isso permite a mistura de imagens naturais e sintéticas, além de outros tipos de mídia, por exemplo, um meteorologista parado na frente de um mapa do tempo. Com essa estrutura, é fácil permitir que os programas interajam com os dados do filme. O segundo, lançado em 2003, é conhecido como **H.264** ou **AVC (Advanced Video Coding)**. Seu objetivo é codificar o vídeo na metade da taxa dos codificadores anteriores para gerar o mesmo nível de qualidade, para dar melhor suporte à transmissão de vídeo pelas redes. Esse codificador é usado para HDTV na maioria dos discos Blu-ray.

Os detalhes de todos esses padrões são muitos e variados. Os padrões mais recentes também têm muito mais recursos e opções de codificação do que os padrões mais antigos. Porém, não entraremos em detalhes. Em sua maior parte, os ganhos na compressão de vídeo com o tempo vieram de várias melhorias pequenas, em vez de mudanças fundamentais no modo como o vídeo é comprimido. Assim, vamos esboçar os conceitos gerais.

MPEG comprime tanto áudio quanto vídeo. Como os codificadores de áudio e vídeo funcionam independentemente, existe a questão de como os dois streamings são sincronizados no receptor. A solução é ter um único clock que gera registros de tempo (timestamps) referentes à hora atual para os dois codificadores. Esses períodos de tempo são incluídos na saída codificada e propagados até o receptor, que pode usá-los para sincronizar os streamings de áudio e vídeo.

A compactação de vídeo MPEG tira proveito de dois tipos de redundâncias que existem nos filmes: espacial e temporal. A redundância espacial pode ser utilizada simplesmente codificando cada quadro separadamente com JPEG. Essa técnica ocasionalmente é utilizada, especialmente quando o acesso aleatório a cada quadro é necessário, como na edição de produções de vídeo. Nesse modo, são alcançados níveis de compressão do JPEG.

Uma compressão adicional pode ser obtida aproveitando-se o fato de que quadros consecutivos frequentemente são quase idênticos. O efeito é menor do que parece a princípio, pois muitos cineastas efetuam cortes entre cenas a cada 3 ou 4 segundos (faça a cronometragem de um filme e conte as cenas). Contudo, até mesmo uma série de 75 quadros muito semelhantes oferece uma boa redução potencial em comparação com a simples codificação de cada quadro separado com JPEG.

Para cenas em que a câmera e o fundo da cena permanecem estáticos e um ou dois atores se movimentam lentamente, quase todos os pixels serão idênticos de um quadro para outro. Nesse caso, subtrair cada quadro de seu antecessor e efetuar a compressão JPEG sobre a diferença funcionaria muito bem. Entretanto, para cenas nas quais a câmera faz uma tomada panorâmica ou muda rapidamente de plano, essa técnica é insuficiente. É necessária alguma forma de compensar esse movimento. Isso é exatamente o que o MPEG faz; essa também é a diferença mais importante entre o MPEG e o JPEG.

A saída do MPEG consiste em três tipos de quadros:

1. quadros I (Intracoded): Imagens estáticas, autocontidas e comprimidas;
2. quadros P (Predictive): Diferença bloco a bloco em relação ao quadro anterior;
3. quadros B (Bidirectional): Diferenças bloco a bloco entre o quadro anterior e quadros futuros.

Os quadros I são apenas imagens estáticas codificadas com JPEG ou algo semelhante. É necessário que os quadros I apareçam periodicamente no streaming de saída (por exemplo, uma ou duas vezes por segundo) por três motivos. Primeiro, o MPEG pode ser usado para uma transmissão por multicast, com os espectadores ajustando a transmissão à vontade. Se todos os quadros dependessem de seus antecessores para voltar ao primeiro, quem perdesse o primeiro quadro nunca poderia decodificar nenhum quadro subsequente. Em segundo lugar, se qualquer quadro fosse recebido com erro, a decodificação adicional não seria mais possível: tudo a partir daí seria lixo. Em terceiro lugar, sem os quadros I, enquanto estivesse executando um avanço ou retrocesso rápido, o decodificador teria de calcular todos os quadros exibidos, de modo a saber o valor total do quadro em que parou.

Ao contrário, os quadros P codificam as diferenças entre os quadros. Eles se baseiam na ideia de **macroblocos**, que cobrem, por exemplo, 16 × 16 pixels em espaço de luminescência e 8 × 8 pixels em espaço de crominância. Um macrobloco é codificado pesquisando-se o quadro anterior em busca dele ou de algo apenas um pouco diferente dele.

Um exemplo de onde os quadros P seriam úteis é mostrado na Figura 7.23. Nela, vemos três quadros consecutivos que têm o mesmo fundo de cena, mas são diferentes na posição de uma pessoa. Os macroblocos contendo o fundo de cena serão exatamente iguais, mas os macroblocos contendo a pessoa terão um deslocamento na posição, medido por algum valor desconhecido, e terão de ser localizados.

O padrão MPEG-1 não especifica como, até onde pesquisar, nem determina a proximidade entre os valores para que a comparação seja considerada válida. Isso depende de cada implementação. Por exemplo, uma implementação pode pesquisar um macrobloco na posição atual do quadro anterior e em todas as outras posições deslocadas $\pm \Delta x$ na direção x e $\pm \Delta y$ na direção y. Para cada posição, o número de correspondências na matriz de luminescência pode ser calculado. A posição com o maior número de pontos deve ser declarada vencedora, desde que permaneça acima do limiar previamente determinado. Caso contrário, o macrobloco seria declarado ausente. Logicamente, algoritmos muito mais sofisticados também são possíveis.

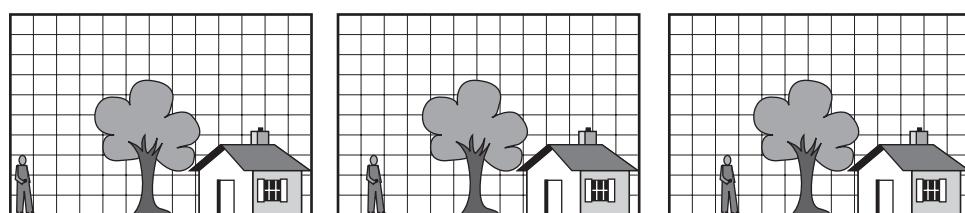



Figura 7.23 | Três quadros consecutivos.

Se um macrobloco for encontrado, ele será codificado verificando-se a diferença em relação a seu valor no quadro anterior (para a luminescência e ambas as crominâncias). A essas matrizes de diferenças é aplicada a transformação cosseno discreta, quantização, codificação run-length e codificação de Huffman, como de costume. O valor para o macrobloco no streaming de saída é então o vetor de movimento (ou seja, a distância que o macrobloco se deslocou a partir de sua posição anterior em cada direção), seguido pela codificação de sua diferença. Se o macrobloco não for localizado no quadro anterior, o valor atual será codificado, assim como em um quadro I.

É obvio que esse algoritmo é altamente assimétrico. Uma implementação é livre para experimentar todas as posições plausíveis no quadro anterior, em uma tentativa desesperada de localizar todos os últimos macroblocos, não importando para onde eles se deslocaram. Essa estratégia minimizaria o streaming de codificação MPEG à custa de uma codificação muito lenta. Essa solução deve ser boa para uma codificação que só ocorra uma vez, como a codificação de uma filmoteca, mas seria terrível para videoconferências em tempo real.

Da mesma forma, cada implementação é livre para decidir o que constitui um macrobloco 'encontrado'. Essa liberdade permite aos implementadores competir pela qualidade e pela velocidade de seus algoritmos, mas sempre produz algoritmos que obedecem à saída MPEG.

Até agora, a decodificação MPEG-1 tem se mostrado simples. Decodificar quadros I é o mesmo que decodificar imagens JPEG. Decodificar quadros P exige que o decodificador armazene o quadro anterior em um buffer e depois construa o novo quadro em um segundo buffer baseado em macroblocos completamente codificados e macroblocos contendo diferenças em relação aos quadros anteriores. O novo quadro é montado a partir de um macrobloco por vez.

Os quadros B são semelhantes aos quadros P, exceto pelo fato de permitirem que o macrobloco de referência esteja em um quadro anterior ou em um quadro seguinte. Essa liberdade adicional acarreta uma melhoria na compensação de movimento, e também é útil quando objetos passam pela frente ou por trás de outros objetos. Para realizar a codificação de quadro B, o codificador precisa manter uma sequência de quadros na memória ao mesmo tempo: o quadro anterior, o atual e o próximo. A decodificação também é mais complicada e gera algum atraso. Isso porque determinado quadro B não pode ser decodificado até que os quadros sucessivos dos quais ele depende sejam decodificados. Assim, ainda que os quadros B ofereçam a melhor compressão, eles nem sempre são usados, devido à sua maior complexidade e requisitos de buffering.

Os padrões MPEG contêm muitas melhorias a essas técnicas, para alcançar excelentes níveis de compressão. AVC pode ser usado para comprimir vídeo em taxas superiores a 50:1, o que reduz os requisitos de largura de banda da rede pelo mesmo fator. Para obter mais informações sobre AVC, consulte Sullivan e Wiegand (2005).

7.4.3 Streaming de mídia armazenada

Agora, vamos prosseguir para as aplicações em rede. Nosso primeiro caso é o streaming de mídia que já está armazenada em arquivos. O exemplo mais comum disso é assistir a vídeos pela Internet. Essa é uma forma de vídeo por demanda, ou **VoD** (**Video on Demand**). Outras formas de vídeo por demanda utilizam uma rede com provedor (separada da Internet) para oferecer os filmes — por exemplo, a rede a cabo.

Na próxima seção, veremos o streaming de mídia ao vivo, por exemplo, o IPTV de broadcast e o rádio pela Internet. Depois, veremos o terceiro caso de conferência em tempo real. Um exemplo é uma chamada de VoIP, ou videoconferência com Skype. Esses três casos impõem requisitos cada vez mais rigorosos sobre o modo como entregamos áudio e vídeo pela rede, pois precisamos prestar mais atenção no atraso e no jitter.

A Internet está repleta de sites de música e vídeo, que fornecem arquivos de mídia armazenada. Na realidade, o modo mais fácil de lidar com a mídia armazenada é *não* enviá-la por streaming. Imagine que você queira criar um site de aluguel de filmes on-line para competir com o iTunes da Apple. Um site normal permitirá que os usuários baixem e depois assistam a vídeos (depois de pagar, é claro). A sequência de etapas aparece na Figura 7.24. Vamos explicar esses passos para compará-los com o próximo exemplo.

O navegador entra em ação quando o usuário clica em um filme. Na etapa 1, ele envia uma solicitação HTTP do filme para o servidor Web ao qual o filme está vinculado.

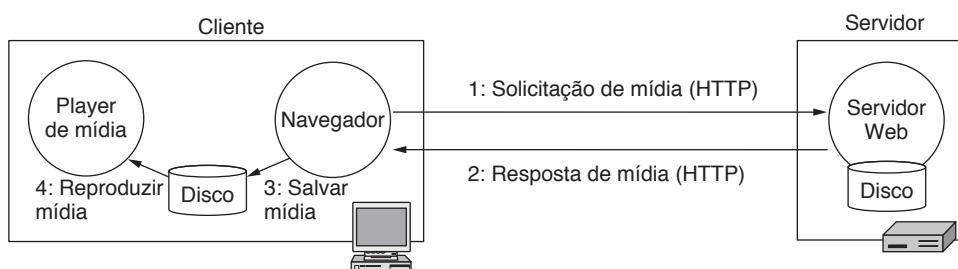



Figura 7.24 | Reproduzindo mídia pela Web através de downloads simples.

Na etapa 2, o servidor busca o filme (que é apenas um arquivo em formato MP4 ou algum outro) e o envia de volta ao navegador. Usando o tipo MIME, por exemplo, *video/mp4*, o navegador pesquisa como deverá exibir o arquivo. Nesse caso, é com um player de mídia que aparece como uma aplicação auxiliar, embora também pudesse ser um plug-in. O navegador salva o filme inteiro em um arquivo auxiliar no disco, na etapa 3. Depois, ele inicia o player de mídia, passando-lhe o nome do arquivo auxiliar. Por fim, na etapa 4, o player de mídia começa a ler o arquivo e reproduzir o filme.

Em princípio, essa técnica é completamente correta. Ela exibirá o filme. Não existe nenhum problema de rede em tempo real a ser resolvido, pois o download é simplesmente um download de arquivo. O único problema é que o vídeo inteiro precisa ser transmitido pela rede antes que o filme comece. A maioria dos clientes não deseja esperar uma hora por seu 'vídeo por demanda'. Esse modelo pode ser problemático até mesmo para o áudio. Imagine a prévia de uma música antes de comprar um disco. Se tiver 4 MB, o que é um tamanho comum para uma música em MP3, e a conectividade de banda larga de 1 Mbps, o usuário ficará esperando com meio minuto de silêncio antes que a prévia comece. Esse modelo provavelmente não venderá muitos discos.

Para contornar esse problema sem alterar o modo como o navegador funciona, os sites podem usar o projeto mostrado na Figura 7.25. A página vinculada ao filme não é o arquivo de filme real. Em vez disso, é o que chamamos de **meta-arquivo**, um arquivo muito pequeno apenas dando nome ao filme (e possivelmente contendo outros descritores importantes). Um meta-arquivo simples poderia ser apenas uma linha de texto ASCII, parecida com esta:

```
rtsp://joes-movie-server/movie-0025.mp4
```

O navegador recebe a página normalmente, agora um arquivo on-line, nas etapas 1 e 2. Depois, ele inicia o player de mídia e lhe entrega o arquivo on-line na etapa 3, como sempre. O player de mídia lê o meta-arquivo e vê o URL de onde deve obter o filme. Ele entra em contato com *joes-movie-server* e pede o filme na etapa 4. O filme é então enviado para o player de mídia na etapa 5. A vantagem desse arranjo é que o player de mídia começa rapidamente, somente após um meta-arquivo muito pequeno ser baixado. Quando isso acontece, o navegador não está mais no loop. A mídia é enviada diretamente para o player, que pode começar a mostrar o filme antes que o arquivo inteiro seja baixado.

Mostramos dois servidores na Figura 7.25, pois o servidor indicado no meta-arquivo normalmente não é o mesmo que o servidor Web. De fato, geralmente, ele nem sequer é um servidor HTTP, mas um servidor de mídia especializado. Nesse exemplo, o servidor de mídia utiliza **RTSP** (**Real Time Streaming Protocol**), conforme indicado pelo nome do esquema *rtsp*.

O player de mídia tem quatro tarefas principais a fazer:

1. administrar a interface com o usuário;
2. tratar dos erros de transmissão;
3. descomprimir o conteúdo;
4. eliminar o jitter.

A maioria dos players de mídia hoje em dia tem uma interface brilhante com o usuário, às vezes simulando uma unidade estéreo, com botões, controles deslizantes e atrativos visuais. Normalmente, existem painéis frontais intercambiáveis, chamados **skins**, que o usuário pode aplicar no player. O player de mídia precisa administrar tudo isso e interagir com o usuário.

As outras tarefas estão relacionadas e dependem dos protocolos de rede. Analisaremos cada uma por vez, começando com o tratamento dos erros de transmissão. Lidar com erros depende da seguinte questão: se o transporte

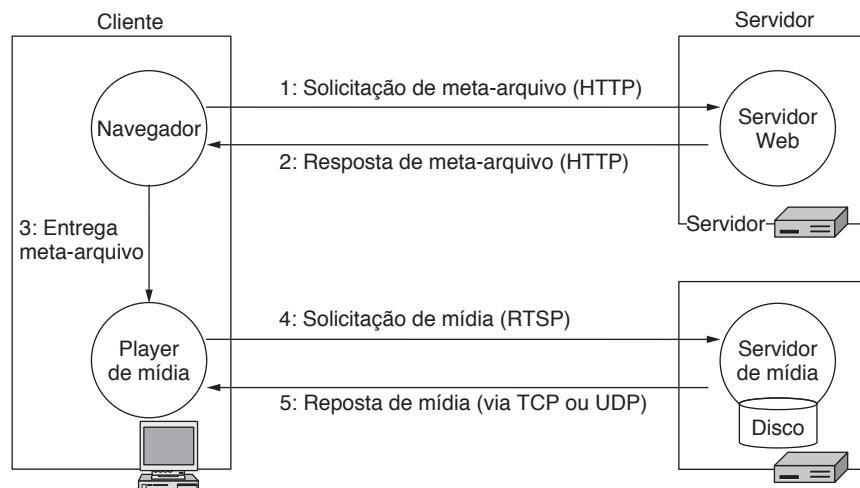



Figura 7.25 | Streaming de mídia usando a Web e um servidor de mídia.

baseado em TCP, como HTTP, é usado para transportar a mídia, ou então é usado um transporte baseado em UDP, como o RTP. Ambos são usados na prática. Se um transporte baseado em TCP estiver sendo utilizado, então não existem erros para o player de mídia corrigir, pois o TCP já oferece confiabilidade usando retransmissões. Esse é um modo fácil de lidar com erros, pelo menos para o player, mas isso complica a remoção do jitter em uma etapa posterior.

Como alternativa, um transporte baseado em UDP, como o RTP, pode ser usado para mover os dados. Estudamos isso no Capítulo 6. Com esses protocolos, não existem retransmissões. Assim, a perda de pacotes devida a congestionamento ou erros de transmissão significará que parte da mídia não chega. O player de mídia fica encarregado de lidar com esse problema.

Vamos entender a dificuldade que precisamos enfrentar. A perda é um problema, pois os clientes não gostam de grandes intervalos em suas músicas ou filmes. Porém, esse não é um problema tão grande quanto a perda em uma transferência de arquivo normal, pois a perda de uma pequena quantidade de mídia precisa não degradar a apresentação para o usuário. Para o vídeo, o usuário provavelmente nem notará que talvez existam 24 novos quadros em algum segundo, em vez de 25 novos quadros. Para o áudio, pequenas lacunas na reprodução podem ser mascaradas com sons próximos no tempo. É provável que o usuário não detecte essa substituição, a menos que esteja prestando *muita* atenção.

Contudo, a chave para esse raciocínio é que as lacunas são muito curtas. O congestionamento da rede ou um erro de transmissão geralmente farão com que um pacote inteiro se perca, e os pacotes normalmente se perdem em pequenas rajadas. Duas estratégias podem ser usadas para reduzir o impacto da perda de pacotes sobre a mídia que é perdida: FEC e intercalação. Vamos descrever cada uma por vez.

FEC (**Forward Error Correction**) é simplesmente a codificação de correção de erro que estudamos no Capítulo 3 aplicada no nível da aplicação. A paridade entre os pacotes oferece um exemplo (Shacham e McKenny, 1990). Para cada quatro pacotes de dados enviados, um quinto **pacote de paridade** pode ser construído e enviado. Isso aparece na Figura 7.26 com os pacotes *A*, *B*, *C* e *D*. O pacote de paridade, *P*, contém bits redundantes que são a paridade ou as somas XOR dos bits em cada um dos quatro pacotes de dados. Com sorte, todos os pacotes chegarão para a maioria dos grupos de cinco pacotes. Quando isso acontece, o pacote de paridade é simplesmente descartado no receptor. Ou, então, se apenas o pacote de paridade se perder, nenhum prejuízo ocorrerá.

Ocasionalmente, porém, um pacote de dados pode ser perdido durante a transmissão, como acontece com *B* na Figura 7.26. O player de mídia recebe apenas três pacotes de dados, *A*, *C* e *D*, mais um de paridade, *P*. Por projeto, os bits no pacote de dados que falta podem ser reconstruídos a partir dos bits de paridade. Para ser específico, usando '+' para representar o XOR ou adição de módulo 2, *B* pode ser reconstruído como $B = P + A + C + D$, pelas propriedades do XOR (ou seja, $X + Y + Y = X$).

FEC pode reduzir o nível de perda visto pelo player de mídia reparando algumas das perdas de pacotes, mas isso só funciona até certo nível. Se dois pacotes em um grupo de cinco se perderem, nada poderemos fazer para recuperar os dados. A outra propriedade a ser observada sobre FEC é o custo que pagamos para obter essa proteção. Cada quatro pacotes se tornam cinco pacotes, de modo que os requisitos de largura de banda para a mídia são 25 por cento maiores. A latência da decodificação também aumenta, pois podemos precisar esperar até que o pacote de paridade tenha chegado antes de podermos reconstruir um pacote de dados que veio antes dele.

Há também um truque inteligente na técnica explicada. No Capítulo 3, descrevemos a paridade como oferecendo detecção de erro. Aqui, estamos oferecendo correção de erro. Como os dois podem ser feitos? A resposta é: neste caso, sabe-se qual pacote foi perdido. Os dados perdidos são chamados de **apagamento**. No Capítulo 3, quando consideramos um quadro que foi recebido com alguns bits com erro, não sabíamos qual bit estava errado. Esse caso é mais difícil de lidar do que os apagamentos. Assim, com eles, a paridade pode oferecer correção de erro, e sem eles a paridade só pode oferecer detecção de erro. Logo, veremos outro benefício inesperado da paridade, quando chegarmos aos cenários de multicast.

A segunda estratégia é chamada **intercalação**. Essa técnica é baseada na mistura ou intercalação da ordem da mídia antes da transmissão e na operação inversa na recep-

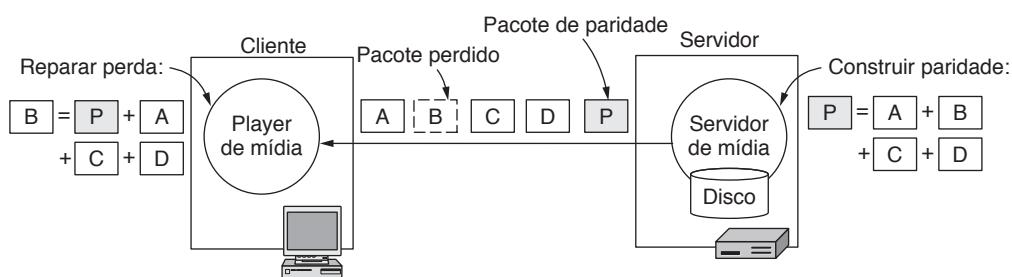



Figura 7.26 | Usando um pacote de paridade para reparar a perda.

ção. Desse modo, se um pacote (ou uma rajada de pacotes) se perder, a perda será espalhada com o tempo pelo processo inverso da mistura. Isso não resultará em uma única lacuna grande quando a mídia for reproduzida. Por exemplo, um pacote poderia conter 220 amostras estéreo, cada uma contendo um par de números de 16 bits, normalmente válidos por 5 ms de música. Se as amostras fossem enviadas em ordem, um pacote perdido representaria uma lacuna de 5 ms na música. Em vez disso, as amostras são transmitidas como na Figura 7.27. Todas as amostras pares para um intervalo de 10 ms são enviadas em um pacote, seguido por todas as amostras ímpares no próximo pacote. A perda do pacote 3 agora não representa uma lacuna de 5 ms na música, mas a perda de cada amostra alternada por 10 ms. Essa perda pode ser tratada facilmente fazendo-se com que o player interpole usando as amostras anterior e seguinte. O resultado é menor resolução temporal para 10 ms, mas não uma lacuna de tempo observável na mídia.

Esse esquema de intercalação só funciona com a amostragem não comprimida. Porém, a intercalação (por períodos de tempo curtos, e não amostras individuais) também pode ser aplicada após a compactação, desde que haja um modo de encontrar limites da amostra no streaming comprimido. A RFC 3119 oferece um esquema que funciona com áudio compactado.

A intercalação é uma técnica atraente quando pode ser usada, pois não precisa de largura de banda adicional, diferentemente de FEC. Porém, a intercalação aumenta a latência, assim como FEC, devido à necessidade de esperar que um grupo de pacotes chegue (de modo que possam ser desintercalados).

A terceira tarefa do player de mídia é descomprimir o conteúdo. Embora essa tarefa seja computacionalmente intensa, ela é bem simples. O problema mais complicado é decodificar a mídia se o protocolo de rede não corrigir erros de transmissão. Em muitos esquemas de compressão, os dados seguintes não poderão ser descomprimidos até que os anteriores o sejam, pois os últimos são codificados em relação aos dados anteriores. Para um transporte baseado em UDP, pode haver perda de pacote. Assim, o processo de codificação precisa ser projetado para permitir a decodificação apesar da perda de pacotes. Esse requisito é o motivo pelo qual MPEG usa quadros I, P e B. Cada quadro I pode ser decodificado independentemente dos outros, para se recuperar da perda de quaisquer quadros anteriores.

A quarta tarefa é eliminar o jitter, a maldição de todos os sistemas em tempo real. A solução geral que descrevemos na Seção 6.4.3 é usar o buffer de reprodução. Todos os sistemas de streaming começam mantendo 5 a 10 segundos de mídia antes de iniciar a reprodução, como mostra a Figura 7.28. A reprodução captura a mídia regularmente do buffer, de modo que o áudio seja claro e o vídeo seja suave. O atraso no início dá ao buffer uma chance de preencher até o limite inferior. A ideia é que os dados agora devem chegar com regularidade suficiente para que o buffer nunca seja completamente esvaziado. Se isso acontecesse, a reprodução de mídia pararia. O valor do buffering é que, se os dados às vezes atrasarem para chegar devido ao congestionamento, a mídia em buffer permitirá que a reprodução continue normalmente, até que a mídia chegue e o buffer seja reposto.

A quantidade de buffering necessária e a velocidade com que o servidor de mídia envia mídia para preencher o buffer dependem dos protocolos de rede. Existem muitas possibilidades. O maior fator no projeto é o uso de um transporte baseado em UDP ou um transporte baseado em TCP.

Suponha que seja usado um transporte baseado em UDP, como RTP. Suponha ainda que haja muita largura de banda para enviar pacotes pelo servidor de mídia para o player de mídia com pouca perda, e pouco tráfego subsequente na rede. Nesse caso, os pacotes podem ser enviados na velocidade exata em que a mídia está sendo reproduzida. Cada pacote transitará pela rede e, depois de um atraso de propagação, chegará praticamente no momento certo para o player de mídia apresentar a mídia. Muito pouco buffering será necessário, pois não existe variabilidade no atraso. Se for usada intercalação ou FEC, mais buffering será necessário, pelo menos para o grupo de pacotes sobre o qual a intercalação ou FEC é realizada. Contudo, isso aumenta apenas uma pequena quantidade de buffering.

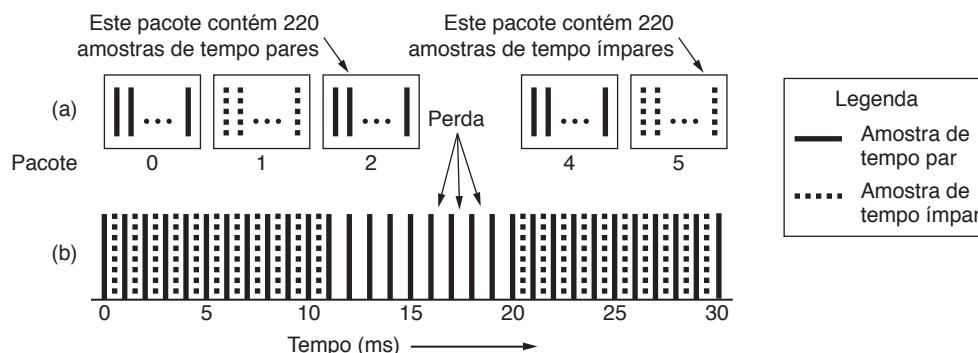



Figura 7.27 | Quando os pacotes transportam amostras alternadas, a perda de um pacote reduz a resolução temporal, em vez de criar uma lacuna no tempo.

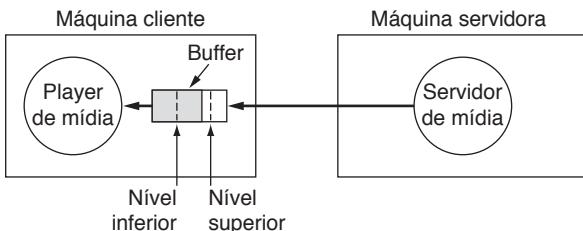



Figura 7.28 | O player de mídia mantém a entrada em buffer, vinda do servidor de mídia, e reproduz a partir do buffer, e não diretamente da rede.

Infelizmente, esse cenário é irreal em dois aspectos. Primeiro, a largura de banda varia pelos caminhos da rede, de modo que normalmente não é claro para o servidor de mídia se haverá largura de banda suficiente antes que ele tente enviar o streaming de mídia. Uma solução simples é codificar a mídia em várias resoluções e permitir que cada usuário escolha uma resolução que seja aceita por sua conectividade com a Internet. Normalmente, existem dois níveis: alta qualidade, digamos, codificada a 1,5 Mbps ou mais, e baixa qualidade, digamos, codificada a 512 kbps ou menos.

Em segundo lugar, haverá algum jitter, ou variação no tempo gasto para as amostras de mídia atravessarem a rede. Esse jitter vem de duas fontes. Normalmente, existe uma quantidade apreciável de tráfego concorrente na rede — parte dele podendo vir da multitarefa dos próprios usuários navegando pela Web enquanto ostensivamente assistem ao streaming do filme. Esse tráfego causará flutuações no tempo em que a mídia chega. Além do mais, estamos preocupados com a chegada de quadros de vídeo e amostras de áudio, e não de pacotes. Com a compressão, os quadros de vídeo em particular podem ser maiores ou menores, dependendo do seu conteúdo. Uma sequência de ação normalmente usará mais bits para codificar do que um panorama ameno. Se a largura de banda da rede for constante, a taxa de entrega de mídia *versus* o tempo variará. Quanto mais jitter, ou variação no atraso, a partir dessas origens, maior precisará ser o nível inferior do buffer para evitar underrun.

Agora, suponha que um transporte baseado em TCP, como HTTP, seja usado para enviar a mídia. Realizando retransmissões e esperando entregar pacotes até que estejam na ordem, o TCP aumentará o jitter que é observado pelo player de mídia, talvez de modo significativo. O resultado é que um buffer maior e um nível superior são necessários. Porém, existe uma vantagem. O TCP enviará dados o mais rápido que a rede puder transportá-los. Às vezes, a mídia será atrasada se a perda tiver que ser reparada. Porém, na maior parte do tempo, a rede poderá entregar a mídia mais rápido do que o player a consome. Nesses períodos, o buffer se encherá e impedirá futuros underruns. Se a rede for significativamente mais rápida do que a taxa média da mídia, como normalmente acontece, o buffer se encherá rapidamente após o início, de modo que seu esvaziamento logo deixará de ser um problema.

Com TCP, ou com UDP e uma taxa de transmissão que excede a taxa de reprodução, uma questão é quanto adiante do ponto de reprodução o player de mídia e o servidor de mídia estão dispostos a prosseguir. Normalmente, eles estão dispostos a baixar o arquivo inteiro.

Entretanto, prosseguir muito além do ponto de reprodução é um trabalho que ainda não parece necessário, pode exigir armazenamento significativo e não é necessário para evitar underruns de buffer. Quando isso não é desejado, a solução é que o player de mídia defina um **nível superior** no buffer. Basicamente, o servidor apenas envia dados até que o buffer esteja cheio, ao nível superior. Depois, o player de mídia lhe pede para interromper. Como os dados continuarão a surgir até que o servidor tenha recebido a solicitação para parar, a distância entre o nível superior e o final do buffer precisa ser maior do que o produto atraso-largura de banda da rede. Depois que o servidor tiver parado, o buffer começará a se esvaziar. Quando ele alcançar o nível inferior, o player de mídia dirá ao servidor de mídia para começar novamente. Para evitar underrun, o nível inferior também precisa levar em conta o produto atraso-largura de banda da rede ao solicitar que o servidor de mídia continue a enviar a mídia.

Para começar e parar o streaming de mídia, o player precisa de um controle remoto. É o que o RTSP oferece. Ele é definido na RFC 2326 e oferece o mecanismo para o player controlar o servidor. Assim como iniciar e interromper o streaming, ele pode recuar ou avançar até uma posição, reproduzir intervalos especificados e reproduzir em velocidades rápidas ou lentas. Isso, porém, não impede o streaming de dados, que normalmente é RTP sobre UDP, ou RTP sobre HTTP (sobre TCP).

Os principais comandos fornecidos pelo RTSP aparecem na Tabela 7.16. Eles têm um formato de texto simples, como mensagens HTTP, e normalmente são transportados por TCP. RTSP também pode ser executado sobre UDP, pois cada comando é confirmado (e portanto pode ser reenviado se não for confirmado).

Embora o TCP pareça não se encaixar bem ao tráfego em tempo real, ele normalmente é usado na prática. O motivo principal é que ele é capaz de passar por firewalls

Comando	Ação do servidor
DESCRIBE	Lista parâmetros da mídia
SETUP	Estabelece um canal lógico entre o player e o servidor
PLAY	Começa a enviar dados ao cliente
RECORD	Começa a aceitar dados do cliente
PAUSE	Interrompe temporariamente o envio de dados
TEARDOWN	Libera o canal lógico

Tabela 7.16 | Comandos RTSP do player ao servidor.

com mais facilidade do que UDP, especialmente quando executado pela porta HTTP. A maioria dos administradores configura firewalls para proteger suas redes contra visitantes mal-intencionados. Eles quase sempre permitem que as conexões TCP pela porta remota 80 passem sobre HTTP e tráfego da Web. Bloquear essa porta ocasiona rapidamente insatisfação nos visitantes. Porém, a maioria das outras portas é bloqueada, incluindo para RSTP e RTP, que usam portas 554 e 5005, entre outras. Assim, o modo mais fácil de fazer com que o streaming de mídia atravesse o firewall é fazer com que o site finja ser um servidor HTTP, enviando uma resposta HTTP normal, pelo menos para o firewall.

Também existem algumas outras vantagens no TCP. Por oferecer confiabilidade, o TCP dá ao cliente uma cópia completa da mídia. Isso torna fácil para um usuário retornar a um ponto de reprodução previamente visto sem preocupação com dados perdidos. Finalmente, o TCP manterá em buffer o máximo de mídia possível, com a maior rapidez possível. Quando o espaço em buffer é barato (por exemplo, quando um disco é usado para armazenamento), o player de mídia pode baixar a mídia enquanto o usuário assiste a ela. Quando o download termina, o usuário pode vê-la sem interrupção, mesmo que perca a conectividade. Essa propriedade é útil para dispositivos móveis, pois a conectividade pode mudar rapidamente com a movimentação.

A desvantagem do TCP é a latência inicial aumentada (devido ao início da conexão TCP) e também ao nível de referência inferior mais alto. Contudo, isso raramente é uma grande penalidade, desde que a largura de banda da rede ultrapasse a taxa de mídia por um grande fator.

7.4.4 Streaming de mídia ao vivo

Não apenas os vídeos gravados são tremendamente populares na Web. O streaming de mídia ao vivo também é muito popular. Quando foi possível fornecer áudio e vídeo pela Internet, as estações comerciais de rádio e TV tiveram a ideia de transmitir seu conteúdo pela Internet assim como pelo ar. Não muito tempo depois disso, as estações de faculdades começaram a colocar seus sinais pela Internet. Em seguida, os *alunos* das faculdades iniciaram suas próprias transmissões pela Internet.

Hoje, pessoas e empresas de todos os tamanhos enviam áudio e vídeo ao vivo. A área é um manancial de inovação, à medida que as tecnologias e os padrões evoluem. O streaming de mídia ao vivo é usado em chamadas on-line pelas principais estações de televisão. Isso é chamado **IPTV (IP TeleVision)**. Também é usado na transmissão radiofônica de broadcast, como a BBC. Isso é chamado **rádio via Internet**. Tanto IPTV quanto rádio via Internet alcançam audiências no mundo inteiro para eventos que variam de shows de moda até a Copa do Mundo de futebol e diversas disputas ao vivo. O streaming ao vivo por IP é usado como uma tecnologia dos provedores a cabo, para montar seus próprios sistemas de broadcast. E é muito usado em operações de menor orçamento, desde sites adultos até zoológicos. Com a tecnologia atual, praticamente qualquer um pode iniciar streaming ao vivo rapidamente e com pouca despesa.

Uma técnica para o streaming ao vivo é gravar programas em disco. Os espectadores podem se conectar aos arquivamentos do servidor, puxar qualquer programa e baixá-lo para escutar. Um **podcast** é um episódio recuperado dessa maneira. Para eventos agendados, também é possível armazenar conteúdo logo após ele ser transmitido ao vivo, de modo que o arquivo só estará rodando, digamos, meia hora ou menos após a apresentação ao vivo.

De fato, essa técnica é exatamente a mesma usada para o streaming de mídia que acabamos de discutir. Ela é fácil de fazer, todas as técnicas que já discutimos funcionam para ela e os espectadores podem escolher entre todos os programas arquivados.

Uma técnica diferente é enviar conteúdo ao vivo pela Internet. Os espectadores sintonizam em um streaming de mídia contínuo, assim como ao ligar a televisão. Porém, os players de mídia oferecem os recursos adicionais de permitir que o usuário interrompa ou 'rebobine' a reprodução. A mídia ao vivo continuará a fluir e será mantida em buffer pelo player até que o usuário esteja pronto para ela. Do ponto de vista do navegador, isso se parece exatamente com o caso do streaming de mídia armazenado. Não importa para o player se o conteúdo vem de um arquivo ou se está sendo enviado ao vivo, e normalmente o player não poderá saber (excetuando que não seja possível saltar para a frente em um streaming ao vivo). Dada a semelhança do mecanismo, grande parte da nossa discussão anterior se aplica, mas existem algumas diferenças importantes.

O importante é que ainda é preciso manter um buffer no lado do cliente para reduzir o jitter. De fato, uma quantidade maior de buffering normalmente é necessária para o streaming ao vivo (independentemente da consideração de que o usuário pode interromper a reprodução). Quando estiver fluindo de um arquivo, a mídia pode ser enviada em uma velocidade maior do que a taxa de reprodução. Isso encherá um buffer rapidamente para compensar o jitter da rede (e o player interromperá o fluxo se não quiser manter mais dados em buffer). Ao contrário, o streaming de mídia ao vivo sempre é transmitido exatamente na velocidade em que é gerado, que é a mesma que a velocidade em que é reproduzido. Ele não pode ser enviado mais rápido do que isso. Por conseguinte, o buffer precisa ser grande o suficiente para lidar com toda a gama de jitter da rede. Na prática, um atraso inicial de 10 a 15 segundos normalmente é adequado, de modo que esse não é um grande problema.

A outra diferença importante é que os eventos de streaming ao vivo normalmente têm centenas ou milhares de espectadores simultâneos do mesmo conteúdo. Sob essas circunstâncias, a solução natural para o streaming ao vivo é usar o multicasting. Esse não é o caso para o streaming de

mídia armazenada, pois os usuários normalmente enviam conteúdo diferente a qualquer momento. O streaming para muitos usuários, então, consiste em muitas sessões de streaming individuais que ocorrem ao mesmo tempo.

Um esquema de streaming multicast funciona da seguinte forma: o servidor envia cada pacote de mídia uma vez usando o multicast IP para um endereço de grupo. A rede entrega uma cópia do pacote para cada membro. Todos os clientes que querem receber o fluxo se juntam ao grupo. Eles fazem isso usando IGMP, em vez de enviar uma mensagem RTSP ao servidor de mídia. Isso porque o servidor de mídia já está enviando o streaming ao vivo (exceto antes que o primeiro usuário se conecte). O que é preciso é arranjar para que o streaming seja recebido localmente.

Como o multicast é um serviço de entrega do tipo um-para-muitos, a mídia é transportada em pacotes RTP por UDP. TCP só opera entre um único transmissor e um único receptor. Como UDP não oferece confiabilidade, alguns pacotes podem se perder. Para reduzir o nível de perda de mídia para um nível aceitável, podemos usar FEC e intercalação, como antes.

No caso do FEC, há uma interação benéfica com o multicast que aparece no exemplo de paridade da Figura 7.29. Quando os pacotes são enviados por multicast, diferentes clientes podem perder diferentes pacotes. Por exemplo, o cliente 1 perdeu o pacote *B*, o cliente 2 perdeu o pacote de paridade *P*, o cliente 3 perdeu *D* e o cliente 4 não perdeu pacote algum. Porém, embora três pacotes diferentes sejam perdidos pelos clientes, cada cliente pode recuperar todos os pacotes de dados nesse exemplo. Tudo o que é preciso é que cada cliente não perca mais do que um pacote, qualquer que seja, de modo que o pacote que falta possa ser recuperado por um cálculo de paridade. Nonnenmacher et al. (1997) descrevem como essa ideia pode ser usada para aumentar a confiabilidade.

Para um servidor com um grande número de clientes, o multicast de mídia em pacotes RTP e UDP certamente é o modo de operação mais eficiente. Caso contrário, ele precisa transmitir *N* streamings quando tiver *N* clientes, o que exigirá uma quantidade muito grande de largura de banda da rede para grandes eventos de streaming.

Pode ser uma surpresa para você descobrir que a Internet não trabalha dessa forma na prática. O que costuma acontecer é que cada usuário estabelece uma conexão TCP separada com o servidor, e a mídia é enviada por essa conexão. Para o cliente, isso é o mesmo que enviar a mídia armazenada. E, assim como no streaming de mídia armazenada, existem vários motivos para essa escolha aparentemente fraca.

O primeiro motivo é que o multicast IP não é amplamente disponível na Internet. Alguns ISPs e redes o admitem internamente, mas normalmente não está disponível nas bordas da rede, pois é necessário fluxo de dados a longas distâncias. Os outros motivos são as mesmas vantagens do TCP sobre UDP, conforme discutimos anteriormente. O fluxo com TCP alcançará quase todos os clientes na Internet, principalmente quando disfarçado como HTTP para passar por firewalls, e a remessa de mídia confiável permite que os usuários retrocedam com facilidade.

Porém, há um caso importante em que UDP e multicast podem ser usados para o streaming: dentro da rede de um provedor. Por exemplo, uma empresa de cabo poderia decidir enviar canais de TV para set-top boxes do cliente usando tecnologia IP em vez de transmissões de vídeo tradicionais. O uso do IP para distribuir o vídeo por broadcast em geral é chamado IPTV, conforme discutimos. Como a companhia de cabo tem controle completo de sua própria rede, ela pode prepará-la para dar suporte ao multicast IP e ter largura de banda suficiente para a distribuição baseada em UDP. Tudo isso é invisível ao cliente, pois a tecnologia IP existe dentro do **jardim cercado** do provedor. Ela se parece com TV a

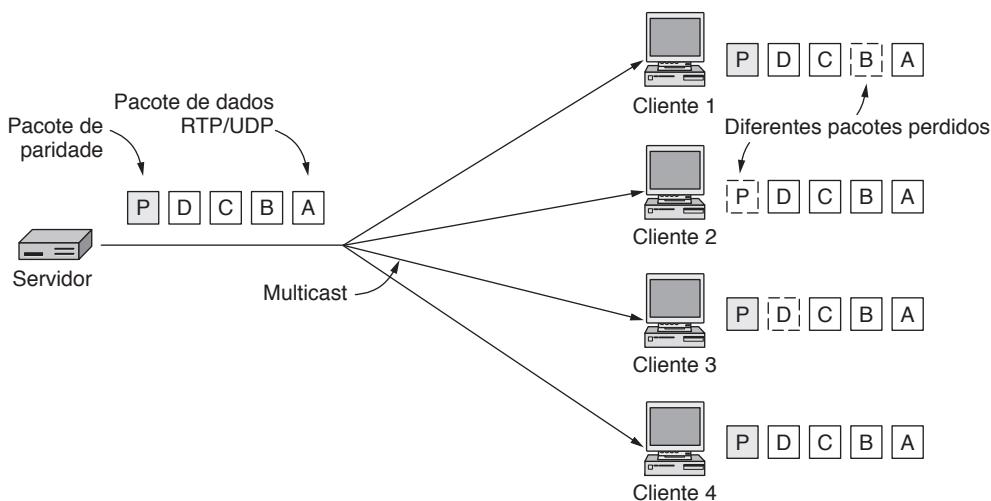



Figura 7.29 | Streaming de mídia por multicast com um pacote de paridade.

cabo em termos de serviço, mas é IP por detrás, com a set-top box sendo um computador rodando UDP e o aparelho de TV sendo simplesmente um monitor conectado ao computador.

De volta ao caso da Internet, a desvantagem do streaming ao vivo sobre o TCP é que o servidor precisa enviar uma cópia separada da mídia para cada cliente. Isso é viável para um número moderado de clientes, especialmente áudio. O truque é colocar o servidor em um local com boa conectividade com a Internet, de modo que haja largura de banda suficiente. Normalmente, isso significa alugar um servidor em um centro de dados a partir de um provedor de hospedagem, e não usar um servidor em casa apenas com conectividade com a Internet por banda larga. Existe um mercado de hospedagem muito competitivo, de modo que isso não precisa ser caro.

De fato, é fácil para qualquer um, até mesmo um estudante, montar e operar um servidor de streaming de mídia, como uma estação de rádio pela Internet. Os principais componentes dessa estação são ilustrados na Figura 7.30. A base da estação é um PC comum com placa de som e microfone decentes. Um software popular é usado para capturar áudio e codificá-lo em vários formatos, por exemplo, MP4, e players de mídia são usados para escutar o áudio normalmente.

O streaming de áudio capturado no PC é então fornecido via Internet a um servidor de mídia com boa conectividade na rede, como podcasts para o streaming do arquivo armazenado ou para o streaming ao vivo. O servidor trata da tarefa de distribuir a mídia por meio de grandes quantidades de conexões TCP. Ele também apresenta um site Web de front-end com páginas sobre a estação e links para o conteúdo que está disponível para o streaming. Existem pacotes de software comerciais para administrar todas essas partes, além de pacotes de código fonte aberto, como icecast.

Contudo, para um número muito grande de clientes, torna-se inviável usar TCP para enviar mídia para cada um a partir de um único servidor. Simplesmente não existe largura de banda suficiente para esse servidor único. Para grandes sites de streaming, o streaming é feito usando um conjunto de servidores espalhados geograficamente, de modo que um cliente pode se conectar ao servidor mais próximo. Essa é uma rede de distribuição de conteúdo, que estudaremos no final do capítulo.

7.4.5 Teleconferência em tempo real

Era uma vez, as chamadas de voz se faziam pela rede telefônica pública comutada, e o tráfego da rede era principalmente voz, com um pouco de tráfego de dados aqui e ali. Depois vieram a Intenet e a Web. O tráfego de dados cresceu mais e mais, até que, por volta de 1999, havia tanto tráfego de dados quanto de voz (como a voz agora é digitalizada, ambos podem ser medidos em bits). Por volta de 2002, o volume do tráfego de dados era uma ordem de grandeza a mais que o volume do tráfego de voz, e ainda crescendo exponencialmente, com o tráfego de voz permanecendo quase estático.

A consequência desse crescimento tem sido a absorção da rede telefônica. O tráfego de voz agora é transportado usando tecnologias de Internet e representa apenas uma pequena fração da largura de banda da rede. Essa tecnologia revolucionária é conhecida como **voz sobre IP**, e também como **telefonia via Internet**.

Voz sobre IP é usada em várias formas, impulsionadas por fortes fatores econômicos. (Em bom português: ela economiza dinheiro, e por isso as pessoas a utilizam.) Uma forma é ter algo que se parece com telefones normais (antiquados?) que se conectam à Ethernet e enviam chamadas pela rede. Pehr Anderson era um aluno cursando o MIT quando, junto de seus amigos, criou um protótipo desse esquema para um projeto de aula. Eles receberam nota 'B'. Não satisfeito, Anderson iniciou uma empresa chamada NBX em 1996, pioneira nesse tipo de voz por IP, e a vendeu para a 3Com por US$ 90 milhões três anos depois. As empresas gostam muito dessa técnica pois lhes permite fugir das linhas telefônicas separadas e ficar com as redes que já possuem.

Outra técnica é usar a tecnologia IP para montar uma rede telefônica de longa distância. Em países como os Estados Unidos, essa rede pode ser acessada pelo serviço competitivo de longa distância, discando para um prefixo específico. As amostras de voz são colocadas em pacotes que

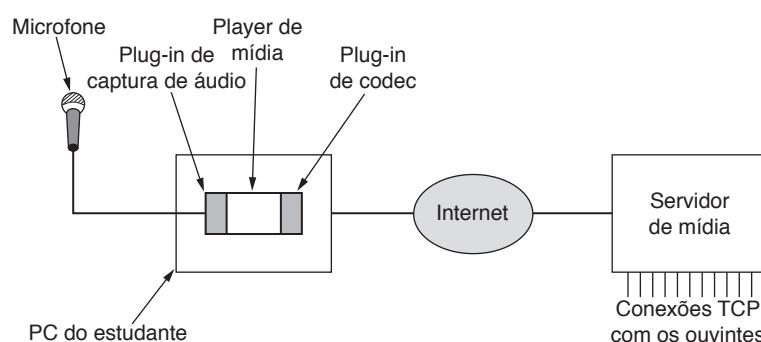



Figura 7.30 | Estação de rádio de um estudante.

são injetados na rede e retirados quando saem dela. Como o equipamento IP é muito mais barato que o equipamento de telecomunicações, isso resulta em serviços mais baratos.

A propósito, a diferença no preço não é totalmente técnica. Por muitas décadas, o serviço telefônico foi um monopólio controlado, o que garantia às companhias telefônicas um lucro porcentual fixo em relação aos seus custos. Não é surpresa que isso as tenha feito aumentar os custos, por exemplo, com muito hardware redundante, justificado em nome da melhor confiabilidade (o sistema telefônico só poderia ficar parado por 2 horas a cada 40 anos, ou 3 minutos por ano em média). Esse efeito era conhecido como a 'síndrome do polo telefônico folheado a ouro'. Desde a desregulamentação, o efeito diminuiu, é claro, mas o equipamento legado ainda existe. A indústria de TI nunca teve nenhuma operação histórica desse tipo, de modo que sempre foi enxuta.

Porém, vamos nos concentrar na forma de voz por IP que provavelmente é a mais visível aos usuários: usar um computador para chamar outro computador. Essa forma tornou-se comum à medida que os PCs começaram a ser vendidos com microfones, alto-falantes, câmeras e CPUs rápidas o suficiente para processar mídia, e as pessoas começaram a se conectar com a Internet em casa por velocidades de banda larga. Um exemplo bem conhecido é o software Skype, que foi lançado a partir de 2003. Skype e outras empresas também oferecem gateways para facilitar a chamada para números de telefone normais, além de computadores com endereços IP.

À medida que a largura de banda aumentou, as chamadas de vídeo se juntaram às chamadas de voz. Inicialmente, as chamadas de vídeo estavam no domínio das empresas. Os sistemas de videoconferência foram projetados para trocar vídeo entre dois ou mais locais, permitindo que executivos em diferentes locais vissem uns aos outros enquanto realizavam suas reuniões. Porém, com boa conectividade de banda larga com a Internet e software de compressão de vídeo, os usuários domésticos também passaram a realizar videoconferência. Ferramentas como Skype, que começaram apenas com áudio, agora normalmente incluem vídeo com as chamadas, para que amigos e família espalhados pelo mundo possam ver e ouvir uns aos outros.

Pelo nosso ponto de vista, chamadas de voz ou vídeo na Internet também são um problema de streaming de mídia, porém muito mais restrito do que o streaming de um arquivo armazenado ou um evento ao vivo. A restrição adicional é a baixa latência necessária para uma conversa bidirecional. A rede telefônica permite uma latência unidirecional de até 150 ms para um uso aceitável, após a qual o atraso começa a incomodar os participantes. (Chamadas internacionais podem ter uma latência de até 400 ms, um ponto em que a experiência do usuário está longe de ser positiva.)

Essa baixa latência é difícil de conseguir. Certamente, o buffering de 5 a 10 segundos de mídia não vai funcionar (como seria para o broadcasting de um evento esportivo ao vivo). Em vez disso, sistemas de vídeo e voz sobre IP precisam ser projetados com uma série de técnicas para minimizar a latência. Esse objetivo significa começar com UDP como uma escolha clara, em vez de TCP, pois as retransmissões do TCP introduzem pelo menos uma ida e volta de atraso. Entretanto, algumas formas de latência não podem ser reduzidas, mesmo com UDP. Por exemplo, a distância entre Seattle e Amsterdã é próxima de 8.000 Km. O atraso de propagação na velocidade da luz para essa distância em fibra óptica é de 40 ms, com muita sorte. Na prática, o atraso de propagação pela rede será maior, pois cobrirá uma distância maior (os bits não seguem uma grande rota circular) e haverá atrasos de transmissão à medida que cada roteador IP armazenar e encaminhar um pacote. Esse atraso fixo é engolido pelo atraso considerado aceitável.

Outra fonte de latência está relacionada ao tamanho do pacote. Em geral, pacotes grandes são a melhor maneira de usar a largura de banda da rede, pois são mais eficientes. Porém, em uma taxa de amostragem de áudio de 64 kbps, um pacote de 1 KB levaria 125 ms para ser preenchido (e ainda mais, se as amostras forem comprimidas). Esse atraso consumiria a maior parte do atraso geral permitido. Além disso, se o pacote de 1 KB for enviado por um enlace com acesso de banda larga que trabalha com apenas 1 Mbps, ele precisará de 8 ms para ser transmitido. Depois, acrescente outros 8 ms para o pacote passar pelo enlace de banda larga na outra extremidade. É claro que pacotes grandes não funcionarão.

Em vez disso, os sistemas de voz sobre IP utilizam pacotes curtos, para reduzir a latência ao custo da eficiência da largura de banda. Eles mantêm amostras de áudio em lotes de unidades menores, normalmente 20 ms. A 64 kbps, isso significa 160 bytes de dados, menos com compressão. Porém, por definição, o atraso dessa divisão em pacotes será de 20 ms. O atraso de transmissão também será menor porque o pacote é mais curto. Em nosso exemplo, ele seria reduzido para algo em torno de 1 ms. Usando pacotes curtos, o atraso mínimo em uma direção para um pacote de Seattle a Amsterdã seria reduzido de um tempo inaceitável de 181 ms (40 + 125 + 16) para um tempo aceitável de 62 ms (40 + 20 + 2).

E ainda nem falamos sobre o overhead do software, mas ele também consome parte do atraso considerado aceitável. Isso é especialmente verdadeiro para o vídeo, pois a compressão costuma ser necessária para incluir o vídeo na largura de banda disponível. Diferentemente do streaming de um arquivo armazenado, não há tempo para haver um codificador computacionalmente intenso para altos níveis de compressão. O codificador e o decodificador precisam trabalhar com muita rapidez.

O buffering ainda é necessário para reproduzir as amostras de mídia em tempo (para evitar áudio ininteligível ou vídeo com paradas), mas a quantidade de buffering precisa ser muito pequena, pois o tempo restante em nosso atraso aceitável é medido em milissegundos. Quando um pacote leva muito tempo para chegar, o player pula as amostras que faltam, talvez reproduzindo o ruído ambiente ou repetindo um quadro para mascarar a perda ao usuário. Há um compromisso entre o tamanho do buffer usado para lidar com o jitter e a quantidade de mídia que é perdida. Um buffer menor reduz a latência, mas resulta em mais perda, devido ao jitter. Como consequência, quando o tamanho do buffer diminui, a perda é observada pelo usuário.

Os leitores atentos podem ter notado que não falamos nada sobre os protocolos da camada de rede até aqui nesta seção. A rede pode reduzir a latência ou, pelo menos, o jitter, usando mecanismos de qualidade de serviço. O motivo para que esse problema não tenha aparecido antes é que o streaming é capaz de operar com latência substancial, mesmo no caso de streaming ao vivo. Se a latência não for um problema importante, um buffer no host final será suficiente para tratar do problema de jitter. Porém, para conferência em tempo real, normalmente é importante fazer com que a rede reduza o atraso e o jitter para ajudar a cumprir o atraso permitido. O único momento em que isso não é importante é quando existe tanta largura de banda da rede disponível que todos recebem um bom serviço.

No Capítulo 5, descrevemos dois mecanismos de qualidade de serviço que ajudam nesse objetivo. Um deles é DS (Differentiated Services), em que os pacotes são marcados como pertencentes a diferentes classes, que recebem tratamento diferente dentro da rede. A marcação apropriada para pacotes de voz sobre IP é baixo atraso. Na prática, os sistemas definem o ponto de código DS para o valor bem conhecido para a classe *Expedited Forwarding* com o tipo de serviço *Low Delay*. Isso é útil em especial para enlaces de acesso de banda larga, que tendem a ficar congestionados quando o tráfego da Web ou outro tráfego compete no uso do enlace. Dado um caminho de rede estável, atraso e jitter são aumentados pelo congestionamento. Cada pacote de 1 KB leva 8 ms para ser enviado por um enlace de 1 Mbps, e um pacote de voz sobre IP trará consigo esses atrasos se tiver que ficar em uma fila atrás do tráfego da Web. Porém, com uma baixa marcação de atraso, os pacotes de voz sobre IP saltarão à frente da fila, adiantando-se aos pacotes Web e reduzindo seu atraso.

O segundo mecanismo que pode reduzir o atraso é garantir que haja largura de banda suficiente. Se a largura de banda disponível variar ou a taxa de transmissão flutuar (como no vídeo compactado) e às vezes não houver largura de banda suficiente, filas serão criadas, o que aumentará o atraso. Isso ocorrerá mesmo com DS. Para garantir uma largura de banda suficiente, uma reserva pode ser feita com a rede. Essa capacidade é fornecida pelos serviços integrados. Infelizmente, isso não é muito implantado. Em vez disso, as redes são projetadas para um nível de tráfego esperado ou os clientes da rede recebem acordos de nível de serviço para determinado nível de tráfego. As aplicações precisam operar abaixo desse nível, para evitar causar congestionamento e introduzir atrasos desnecessários. Para a videoconferência casual em casa, o usuário pode escolher uma qualidade de vídeo como um proxy para necessidades de largura de banda, ou o software pode testar o caminho da rede e selecionar uma qualidade apropriada automaticamente.

Qualquer um desses fatores pode fazer com que a latência fique inaceitável, de modo que a conferência em tempo real requer que se preste atenção em todos eles. Para obter uma visão geral de voz sobre IP e uma análise desses fatores, consulte Goode (2002).

Agora que já discutimos o problema da latência no caminho do streaming de mídia, vamos prosseguir para o outro problema principal que os sistemas de conferência precisam resolver. Esse problema é como estabelecer e encerrar chamadas. Veremos dois protocolos que são muito utilizados para essa finalidade, H.323 e SIP. Skype é outro sistema importante, mas seu funcionamento interno é fechado.

H.323

Ficou claro para todos desde o início que, antes que as chamadas de voz e vídeo fossem feitas pela Internet, se cada fornecedor projetasse sua própria pilha de protocolos, o sistema nunca funcionaria. Para evitar esse problema, várias partes interessadas se reuniram sob o patrocínio da ITU para desenvolver padrões. Em 1996, a ITU emitiu a recomendação **H.323**, intitulada 'Visual Telephone Systems and Equipment for Local Area Networks Which Provide a Non-Guaranteed Quality of Service' (ou seja, sistemas e equipamentos de telefonia visual para redes locais que oferecem uma qualidade de serviço não garantida). Só mesmo a indústria de telefonia pensaria em tal título. A recomendação rapidamente mudou para 'Packet-based Multimedia Communications Systems' na revisão de 1998. H.323 foi a base para os primeiros sistemas amplamente difundidos de conferência da Internet. Ela continua sendo a solução mais utilizada, com sua sétima versão em 2009.

A recomendação H.323 é mais uma avaliação da arquitetura de telefonia da Internet que um protocolo específico. Ela faz referência a um grande número de protocolos específicos para codificação de voz, configuração de chamadas, sinalização, transporte de dados e outras áreas, em vez de especificar propriamente cada um desses elementos. O modelo geral é representado na Figura 7.31. No centro há um **gateway** que conecta a Internet à rede de telefonia. Ele se comunica por meio dos protocolos H.323 no lado da Internet e dos protocolos PSTN no lado da rede telefônica. Os dispositivos de comunicação são chamados **terminais**. Uma LAN pode ter um **gatekeeper** (guardião) que controla os pontos extremos sob sua jurisdição, denominada **zona**.

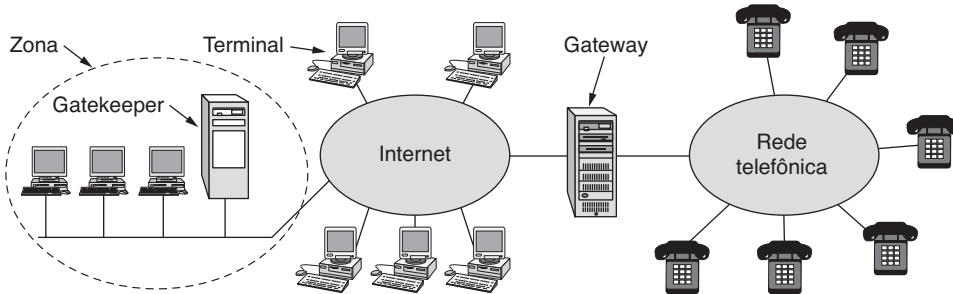



Figura 7.31 | O modelo arquitetônico do H.323 para telefonia da Internet.

Uma rede de telefonia necessita de vários protocolos. Para começar, existe um protocolo para codificação e decodificação de áudio e vídeo. As representações da telefonia-padrão para um único canal de voz como 64 kbps de áudio digital (8 mil amostras de 8 bits por segundo) são definidas na recomendação **G.711** da ITU. Todos os sistemas H.323 devem ter suporte para a G.711. Outras codificações que comprimem a voz são permitidas, mas não exigidas. Os sistemas H.323 empregam diversos algoritmos de compressão e admitem diferentes relações entre qualidade e largura de banda. Para o vídeo, as formas MPEG de compressão de vídeo que descrevemos anteriormente são admitidas, incluindo H.264.

Tendo em vista que são possíveis diversos algoritmos de compactação, é necessário um protocolo para permitir que os terminais negociem o algoritmo que vão usar. Esse protocolo é chamado **H.245**. Ele também negocia outros aspectos da conexão, como a taxa de bits. O RTCP é necessário para controlar os canais do RTP. Também é preciso um protocolo para estabelecer e encerrar conexões, fornecer sinais de discagem, gerar sons de chamada e o restante da telefonia-padrão. A ITU **Q.931** é usada aqui. Os terminais também precisam de um protocolo para se comunicar com o gatekeeper (se existir). Para esse propósito, é usado o **H.225**. O canal do PC para o gatekeeper que ele gerencia é chamado canal **RAS (Registration/Admission/Status)**. Esse canal permite que os terminais entrem e saiam da zona, solicitem e retornem largura de banda e forneçam atualizações de status, entre outras coisas. Por fim, é necessário um protocolo para a transmissão de dados reais. O RTP sobre UDP é usado com esse propósito. Ele é gerenciado pelo RTCP, como sempre. O posicionamento de todos esses protocolos é mostrado na Figura 7.32.

Para ver como esses protocolos funcionam juntos, considere o caso de um terminal de PC em uma LAN (com um gatekeeper) que chama um telefone remoto. Primeiro, o PC tem de descobrir o gatekeeper e, para isso, transmite por broadcast um pacote UDP de descoberta para a porta 1718. Quando responde, o PC descobre o endereço IP do gatekeeper. Agora, o PC se registra com o gatekeeper, enviando a ele uma mensagem RAS em um pacote UDP. Depois que a mensagem é aceita, o PC envia ao gatekeeper uma mensagem RAS de admissão solicitando largura de banda. Só depois que a largura de banda for concedida, será possível iniciar a configuração de chamada. A ideia de solicitar largura de banda com antecedência tem a finalidade de permitir ao gatekeeper limitar o número de chamadas, a fim de evitar saturar a linha de saída, e desse modo oferecer a qualidade de serviço necessária.

A propósito, o sistema telefônico faz a mesma coisa. Quando você pega o receptor, um sinal é enviado à central local. Se a central tiver capacidade de reserva suficiente para outra chamada, ela gerará um sinal de discagem. Se não, você não ouvirá nada. Hoje, o sistema é tão superdimensionado que o tom de discagem é quase sempre instantâneo,

<table>
<tr><td>Áudio</td><td>Vídeo</td><td colspan="4">Controle</td></tr>
<tr><td>G.7xx</td><td>H.26x</td><td rowspan="2">RTCP</td><td rowspan="2">H.225
(RAS)</td><td rowspan="2">Q.931
(sinalização)</td><td rowspan="2">H.245
(controle de
chamada)</td></tr>
<tr><td colspan="2">RTP</td></tr>
<tr><td colspan="4">UDP</td><td colspan="2">TCP</td></tr>
<tr><td colspan="6">IP</td></tr>
<tr><td colspan="6">Protocolo da camada de enlace</td></tr>
<tr><td colspan="6">Protocolo da camada física</td></tr>
</table>

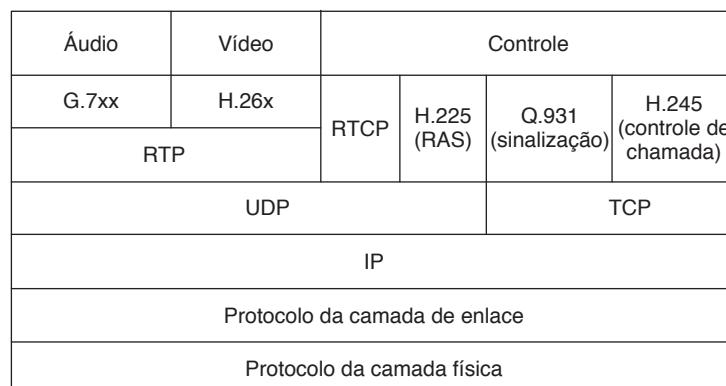



Figura 7.32 | A pilha de protocolos H.323.

mas, nos primeiros dias da telefonia, normalmente isso levava alguns segundos. Assim, se seus netos lhe perguntarem 'por que existem tons de discagem?', agora você já sabe. Até lá, porém, provavelmente os telefones não existirão mais.

Agora, o PC estabelece uma conexão TCP para o gatekeeper, a fim de iniciar a configuração de chamada. A configuração de chamada utiliza os protocolos existentes da rede telefônica, que são orientados a conexões e, portanto, o TCP é necessário. Em contrapartida, o sistema telefônico não tem nada semelhante ao RAS que permita aos telefones anunciar sua presença, e assim os projetistas do H.323 ficaram livres para usar o UDP ou o TCP para RAS. Eles optaram pelo UDP por ter um overhead mais baixo.

Depois que a largura de banda é alocada, o PC pode enviar uma mensagem Q.931 *SETUP* pela conexão TCP. Essa mensagem especifica o número do telefone que está sendo chamado (ou o endereço IP e a porta, se for uma chamada para um computador). O gatekeeper responde com uma mensagem Q.931 *CALL PROCEEDING* para confirmar o recebimento correto da solicitação. Então, o gatekeeper encaminha a mensagem *SETUP* para o gateway.

O gateway, que é metade computador e metade switch de telefonia, faz uma chamada telefônica comum para o telefone desejado (comum). A estação final à qual o telefone está conectado faz soar o sinal do telefone chamado e também envia de volta uma mensagem Q.931 *ALERT* para informar ao PC chamador que a chamada teve início. Quando a pessoa na outra extremidade da linha atende ao telefone, a estação final envia de volta uma mensagem Q.931 *CONNECT* para indicar ao PC que ele tem uma conexão.

Após o estabelecimento da conexão, o gatekeeper não está mais no loop, mas é claro que o gateway está. Os pacotes subsequentes ignoram o gatekeeper e vão diretamente para o endereço IP do gateway. Nesse momento, só temos um canal livre entre as duas partes. Trata-se apenas de uma conexão da camada física para movimentação de bits, e nada mais. Nenhum dos lados conhece detalhe algum sobre o outro.

O protocolo H.245 é usado agora para negociar os parâmetros da chamada. Ele usa o canal de controle H.245, que está sempre aberto. Cada lado começa anunciando seus recursos; por exemplo, se pode lidar com vídeo (o H.323 pode manipular vídeo) ou chamadas de conferência, quais codecs são aceitos etc. Depois que cada lado sabe o que o outro pode manipular, são configurados dois canais de dados unidirecionais, e também são atribuídos a cada parte um codec e outros parâmetros. Tendo em vista que cada lado pode ter um equipamento diferente, é possível que os codecs dos canais direto e reverso sejam diferentes. Depois de concluídas todas as negociações, o fluxo de dados pode começar a usar o RTP. Ele é gerenciado com o RTCP, que desempenha uma função importante no controle de congestionamento. Se houver vídeo presente, o RTCP vai lidar com a sincronização de áudio/vídeo. Os diversos canais são mostrados na Figura 7.33. Quando uma das partes desliga o telefone, o canal de sinalização de chamadas Q.931 é usado para liberar os recursos que não são mais necessários.

Quando a chamada termina, o PC que chama entra em contato com o gatekeeper novamente com uma mensagem RAS para liberar a largura de banda que lhe foi atribuída. Como alternativa, ele pode fazer outra chamada.

Ainda não dissemos nada sobre a qualidade de serviço como parte do H.323, embora tenhamos dito que essa é uma parte importante para tornar a conferência em tempo real um sucesso. O motivo é que QoS está fora do escopo do H.323. Se a rede subjacente for capaz de produzir uma conexão estável, sem jitter, do PC que chama até o gateway, QoS na chamada será boa; caso contrário, não será. Porém, qualquer parte da chamada no lado do telefone estará livre de jitter, pois é assim que a rede telefônica foi projetada.

SIP – Session Initiation Protocol

O H.323 foi projetado pela ITU. Muitas pessoas na comunidade da Internet viam esse protocolo como um produto típico das empresas de telecomunicações: grande, complexo e inflexível. Consequentemente, a IETF estabeleceu um comitê para projetar uma forma mais simples e mais modular de utilizar voz sobre IP. O resultado mais importante até hoje é o **SIP** (**Session Initiation Protocol**). A versão mais recente é descrita na RFC 3261, que foi escrita em 2002. Esse protocolo descreve como preparar chamadas telefônicas via Internet, videoconferências e outras conexões de multimídia. Diferentemente do H.323, que é um conjunto de protocolos completo, o SIP tem um

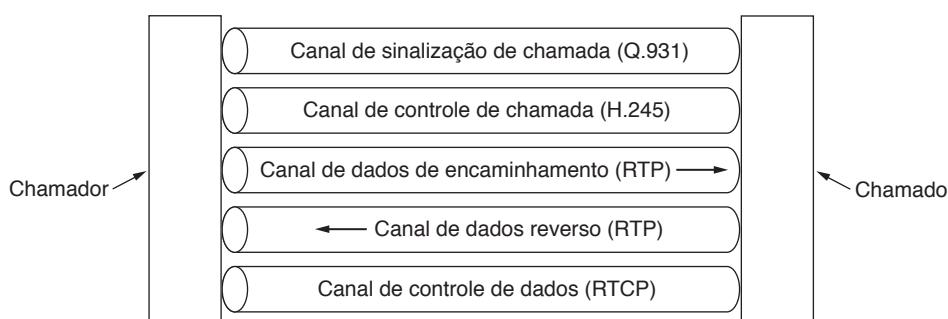



Figura 7.33 | Canais lógicos entre o chamador e o chamado durante a realização de uma chamada.

único módulo, mas foi projetado para interoperar bem com aplicações da Internet existentes. Por exemplo, ele define números de telefones como URLs, de forma que as páginas Web possam conter esses números, permitindo que um clique em um link inicie uma ligação telefônica (da mesma forma que o esquema *mailto* permite que um clique sobre um link abra um programa para enviar uma mensagem de correio eletrônico).

O SIP pode estabelecer sessões de duas partes (ligações telefônicas comuns), sessões de várias partes (em que todos podem ouvir e falar) e sessões de multicast (com um transmissor e muitos receptores). As sessões podem conter áudio, vídeo ou dados, sendo que os dados são úteis para, por exemplo, a realização de jogos em tempo real com vários participantes. O SIP cuida apenas da configuração, do gerenciamento e do encerramento de sessões. Outros protocolos, como RTP/RTCP, são usados para transporte de dados. O SIP é um protocolo da camada de aplicação e pode funcionar sobre o UDP ou o TCP, conforme a necessidade.

O SIP admite uma grande variedade de serviços, inclusive localização de chamada (que pode não estar em sua máquina local) e determinação dos recursos de chamada, bem como tratamento do mecanismo de configuração e encerramento de chamadas. No caso mais simples, o SIP configura uma sessão entre o computador do chamador e o computador chamado; portanto, vamos examinar primeiro esse caso.

Os números de telefones no SIP são representados como URLs que utilizam o esquema *sip*, por exemplo, sip:ilse@cs.universiry.edu, para uma usuária chamada Ilse no host especificado pelo nome DNS cs.university.edu. Os URLs do SIP também podem conter endereços IPv4, endereços IPv6 ou números de telefone reais.

O protocolo SIP é um protocolo de texto modelado sobre o HTTP. Uma parte envia uma mensagem em texto ASCII que consiste em um nome do método na primeira linha, seguido por linhas adicionais contendo cabeçalhos para passagem de parâmetros. Muitos cabeçalhos são tirados do MIME para permitir ao SIP interoperar com aplicações da Internet existentes. Os seis métodos definidos pela especificação do núcleo estão listados na Tabela 7.17.

Para estabelecer uma sessão, o chamador cria uma conexão TCP com o chamado e envia uma mensagem *INVITE* sobre ela, ou então envia a mensagem *INVITE* em um pacote UDP. Em ambos os casos, os cabeçalhos da segunda linha e das linhas subsequentes descrevem a estrutura do corpo da mensagem, que contém os recursos do chamador, tipos de mídia e formatos. Se o chamado aceitar a ligação, ele responderá com um código de resposta do tipo HTTP (um número de três dígitos usando os grupos da Tabela 7.13; 200 significa aceitação). Seguindo a linha do código de resposta, o chamado também pode fornecer informações sobre seus recursos, tipos de mídia e formatos.

A conexão é feita com o uso de um handshake de três vias, de forma que o chamador responde com uma mensagem *ACK* para finalizar o protocolo e confirmar o recebimento da mensagem 200.

Qualquer uma das partes pode solicitar o término de uma sessão enviando uma mensagem que contém o método *BYE*. Quando o outro lado confirmar, a sessão será encerrada.

O método *OPTIONS* é usado para consultar uma máquina sobre seus próprios recursos. Em geral, ele é usado antes de uma sessão ser iniciada, a fim de descobrir se essa máquina é capaz de se comunicar usando voz sobre IP ou se está sendo utilizado qualquer outro tipo de sessão.

O método *REGISTER* se relaciona com a habilidade do SIP em localizar um usuário que está longe de casa e se conectar a ele. Essa mensagem é enviada a um servidor de localização do SIP que controla a localização de cada usuário. Esse servidor pode ser consultado mais tarde para descobrir a localização atual do usuário. A operação de redirecionamento é ilustrada na Figura 7.34. Aqui, o chamador envia a mensagem *INVITE* a um servidor proxy para ocultar o possível redirecionamento. Então, o proxy procura o usuário e envia a mensagem *INVITE* a ele. Em seguida ele atua como um repasse às mensagens subsequentes no handshake de três vias. As mensagens *LOOKUP* e *REPLY* não fazem parte do SIP; qualquer protocolo conveniente pode ser usado, dependendo do tipo de servidor de localização usado.

O SIP tem uma grande variedade de outros recursos que não descreveremos aqui, inclusive a espera de cha-

Método	Descrição
INVITE	Solicita o início de uma sessão
ACK	Confirma que uma sessão foi iniciada
BYE	Solicita o término de uma sessão
OPTIONS	Consulta um host sobre seus recursos
CANCEL	Cancela uma solicitação pendente
REGISTER	Informa um servidor de redirecionamento sobre a localização atual do usuário

Tabela 7.17 | Métodos SIP.

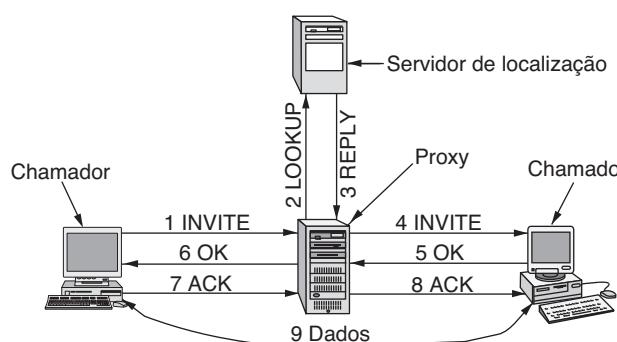



Figura 7.34 | O uso de um proxy e de servidores de redirecionamento com o SIP.

madas, triagem de chamadas, criptografia e autenticação. Ele também tem a capacidade de efetuar chamadas de um computador para um telefone comum, se houver um gateway apropriado disponível entre a Internet e o sistema de telefonia.

Comparação entre H.323 e SIP

O H.323 e o SIP têm muitas semelhanças, mas também apresentam algumas diferenças. Ambos admitem negociação de parâmetros, criptografia e protocolos RTP/RTCP. Um resumo das semelhanças e diferenças entre eles é apresentado na Tabela 7.18.

Embora os conjuntos de características sejam semelhantes, os dois protocolos diferem extensamente na filosofia. O H.323 é um padrão pesado, típico da indústria de telefonia, especificando a pilha de protocolos completa e definindo com precisão o que é permitido e o que é proibido. Essa abordagem leva a protocolos muito bem definidos em cada camada, facilitando a tarefa de interoperabilidade. O preço pago é um padrão grande, complexo e rígido, difícil de adaptar a aplicações futuras.

Ao contrário, o SIP é um protocolo típico da Internet e funciona permutando pequenas linhas de texto ASCII. Esse é um módulo leve que interopera bem com outros protocolos da Internet, mas não muito bem com os protocolos de sinalização do sistema telefônico existente.

Pelo fato de o modelo de voz sobre IP da IETF ser altamente modular, ele é flexível e pode ser adaptado com facilidade a novas aplicações. A desvantagem reside nos problemas potenciais de interoperabilidade, enquanto as pessoas tentam interpretar o que o padrão significa.

7.5 Entrega de conteúdo

Houve uma época em que a Internet era toda comunicação, como a rede telefônica. No início, os acadêmicos se comunicavam com máquinas remotas, fazendo login pela rede para realizar suas tarefas. As pessoas usaram o e-mail para se comunicar umas com as outras por um longo tempo, e agora também usam vídeo e voz sobre IP. Porém, como a Web cresceu, a Internet tornou-se mais relativa a conteúdo do que comunicação. Muitas pessoas usam a Web para buscar informações, e existe uma vasta quantidade de compartilhamento de arquivos peer-to-peer controlada para acesso a filmes, música e programas. A passagem para o conteúdo foi tão pronunciada que a maior parte da largura de banda da Internet agora é usada para entregar vídeos armazenados.

Como a tarefa de distribuir conteúdo é diferente daquela da comunicação, ela impõe diferentes requisitos sobre a rede. Por exemplo, se Sandra quiser falar com Jú-

Item	H.323	SIP
Projetado por	ITU	IETF
Compatibilidade com PSTN	Sim	Ampla
Compatibilidade com a Internet	Sim, com o tempo	Sim
Arquitetura	Monolítica	Modular
Completeza	Pilha de protocolos completa	SIP lida apenas com a configuração
Negociação de parâmetros	Sim	Sim
Sinalização de chamadas	Q.931 sobre TCP	SIP sobre TCP ou UDP
Formato de mensagens	Binário	ASCII
Transporte de mídia	RTP/RTCP	RTP/RTCP
Chamadas de vários participantes	Sim	Sim
Conferências de multimídia	Sim	Não
Endereçamento	URL ou número de telefone	URL
Término de chamadas	Explícito ou encerramento por TCP	Explícito ou por timeout
Mensagens instantâneas	Não	Sim
Criptografia	Sim	Sim
Tamanho do documento de padrões	1.400 páginas	250 páginas
Implementação	Grande e complexa	Moderada, mas há problemas
Status	Distribuído, esp. vídeo	Alternativo, esp. voz

Tabela 7.18 | Comparação H.323 e SIP.

nior, ela pode fazer uma chamada de voz sobre IP para o dispositivo móvel dele. A comunicação precisa ser com um computador em particular; não será possível chamar o computador de Paulo. Mas, se Júnior quiser assistir à última partida de futebol de seu time, ele ficará satisfeito com o streaming de vídeo de qualquer computador que possa oferecer o serviço. Ele não se importa se o computador é o de Sandra ou o de Paulo ou, mais provavelmente, um servidor desconhecido na Internet. Ou seja, o local não importa para o conteúdo, exceto pelo aspecto do desempenho (e da legalidade).

A outra diferença é que alguns sites que oferecem conteúdo se tornaram incrivelmente populares. YouTube é um exemplo clássico. Ele permite que os usuários compartilhem vídeos de sua própria criação, sobre qualquer assunto que se possa imaginar. Muitas pessoas querem fazer isso. O restante de nós quer apenas assistir. Com todos esses vídeos devoradores de largura de banda, estima-se que o YouTube seja responsável por até 10 por cento do tráfego da Internet hoje.

Nenhum servidor isolado é poderoso ou confiável o bastante para lidar com um nível de demanda desse tipo. Em vez disso, o YouTube e outros grandes provedores de conteúdo montam suas próprias redes de distribuição de conteúdo. Essas redes usam centros de dados espalhados pelo mundo inteiro para atender com conteúdo a um número extremamente grande de clientes, com bom desempenho e disponibilidade.

As técnicas que são usadas para a distribuição de conteúdo foram desenvolvidas com o tempo. Desde cedo, com o crescimento da Web, sua popularidade quase foi sua ruína. Mais demandas por conteúdo levaram a servidores e redes que constantemente ficavam sobrecarregados. Muitas pessoas começaram a chamar a WWW de *World Wide Wait* (do inglês, espera de alcance mundial).

Em resposta à demanda do consumidor, quantidades muito grandes de largura de banda foram providas no núcleo da Internet, e a conectividade de banda larga mais rápida foi implantada na borda da rede. Essa largura de banda foi essencial para melhorar o desempenho, mas é apenas parte da solução. Para reduzir os atrasos sem fim, os pesquisadores também desenvolveram diferentes arquiteturas para usar a largura de banda para distribuir conteúdo.

Uma dessas arquiteturas é uma rede de distribuição de conteúdo, ou **CDN** (**Content Distribution Network**). Nela, um provedor estabelece uma coleção distribuída de máquinas em locais dentro da Internet e as utiliza para enviar conteúdo aos clientes. Essa é a escolha das grandes empresas. Uma arquitetura alternativa é uma rede **P2P** (**Peer-to-Peer**). Nela, uma coleção de computadores compartilha seus recursos para atender uns aos outros com conteúdo, sem servidores providos separadamente ou sem ponto de controle central algum. Essa ideia capturou a imaginação das pessoas porque, atuando juntos, muitos pequenos participantes podem conseguir uma boa fatia.

Nesta seção, veremos o problema de distribuir conteúdo na Internet e algumas das soluções que são usadas na prática. Depois de discutir rapidamente sobre a popularidade do conteúdo e o tráfego na Internet, vamos descrever como montar servidores poderosos e usar o caching para melhorar o desempenho para os clientes Web. Depois, entraremos nas duas principais arquiteturas para distribuir conteúdo: CDNs e redes P2P. Como veremos, seu projeto e suas propriedades são muito diferentes.

7.5.1 Conteúdo e tráfego na Internet

Para projetar e preparar redes que funcionam bem, precisamos entender o tráfego que elas precisam transportar. Com a mudança para conteúdo, por exemplo, os servidores migraram de escritórios da empresa para centros de dados da Internet, que oferecem grandes quantidades de máquinas com excelente conectividade de rede. Hoje, para usar até mesmo um servidor pequeno, é mais fácil e mais barato alugar um servidor virtual hospedado em um centro de dados da Internet do que operar uma máquina real em casa ou no escritório, com conectividade de banda larga com a Internet.

Felizmente, existem apenas dois fatos sobre o tráfego da Internet fundamentais a saber. O primeiro fato é que ele muda rapidamente, não apenas nos detalhes, mas na composição geral. Antes de 1994, a maior parte do tráfego era transferência de arquivo FTP tradicional (para mover programas e conjuntos de dados entre os computadores) e e-mail. Depois, a Web chegou e cresceu exponencialmente. O tráfego na Web deixou o tráfego de FTP e e-mail comendo poeira muito antes da 'bolha do ponto com', em 2000. A partir de 2000, o compartilhamento de arquivos P2P para música e depois para filmes decolou. Por volta de 2003, a maior parte do tráfego da Internet era tráfego P2P, deixando a Web comendo poeira. Em algum ponto no fim da década de 2000, o streaming de vídeo usando métodos de distribuição de conteúdo por sites como YouTube começou a ultrapassar o tráfego P2P. A Cisco prevê que, por volta de 2014, 90 por cento de todo o tráfego da Internet seja de vídeo, de uma forma ou de outra (Cisco, 2010).

Nem sempre é o volume de tráfego que importa. Por exemplo, embora o tráfego de voz sobre IP tenha explodido mesmo antes que a Skype fosse iniciada, em 2003, ele sempre será pouca coisa no geral, pois os requisitos de largura de banda do áudio são duas ordens de grandeza menores do que para o vídeo. Porém, o tráfego de voz sobre IP estressa a rede de outras maneiras, pois ele é sensível à latência. Como outro exemplo, as redes sociais on-line cresceram assustadoramente desde que o Facebook foi iniciado, em 2004. Em 2010, pela primeira vez, o Facebook alcançou mais usuários na Web por dia do que o Google. Mesmo colocando o tráfego de lado (e existe realmente muito tráfego), as redes sociais on-line são importantes porque estão mudando o modo como as pessoas interagem por meio da Internet.

A conclusão a que estamos chegando é que mudanças sísmicas no tráfego da Internet acontecem rapidamente, e com certa regularidade. O que virá em seguida? Pergunte novamente na próxima edição deste livro e você saberá.

O segundo fato essencial sobre o tráfego da Internet é que ele é altamente tendencioso. Muitas propriedades com as quais estamos acostumados estão agrupadas em torno de uma média. Por exemplo, a maioria dos adultos está próxima da altura média. Existem algumas pessoas altas e algumas pessoas baixas, mas poucas pessoas muito altas ou muito baixas. Para esses tipos de propriedades, é possível conceber algo para uma faixa que não seja muito grande mas que, apesar disso, captura a maior parte da população.

O tráfego da Internet não é assim. Por muito tempo, soube-se que existe um pequeno número de sites com um tráfego maciço e um grande número de sites com muito pouco tráfego. Essa característica tornou-se parte da linguagem da rede. Os primeiros artigos falavam sobre o tráfego em termos de **trens de pacotes**, com a ideia de que trens expressos com um grande número de pacotes de repente viajariam por um enlace (Jain e Routhier, 1986). Isso foi formalizado como a noção de **autossemelhança**, o que para os nossos propósitos pode ser entendido como o tráfego da rede que exibe muitas lacunas curtas e longas, mesmo quando vistas em diferentes escalas de tempo (Leland et al., 1994). O trabalho mais recente falava de longos fluxos de tráfego como **elefantes** e curtos fluxos de tráfego como **camundongos**. A ideia é que só existem alguns poucos elefantes e muitos camundongos, mas os elefantes são mais importantes porque são muito grandes.

Voltando ao conteúdo da Web, o mesmo tipo de tendência é evidente. A experiência com lojas de aluguel de vídeo, bibliotecas públicas e outras organizações desse tipo mostra que nem todos os itens são igualmente populares. Experimentalmente, quando N filmes estão disponíveis, a fração de todas as solicitações para o k-ésimo mais popular é aproximadamente C/k. Aqui, C é calculado para normalizar a soma como 1, ou seja,

$$C = 1/(1 + 1/2 + 1/3 + 1/4 + 1/5 + \ldots + 1/N)$$

Assim, o filme mais popular é sete vezes mais popular que o filme número sete. Esse resultado é conhecido como **lei de Zipf** (Zipf, 1949). Ela tem o nome de George Zipf, professor de linguística na Universidade de Harvard, que observou que a frequência do uso de uma palavra em um grande corpo de texto é inversamente proporcional à sua classificação. Por exemplo, a 40ª palavra mais comum é usada pelo dobro de vezes que a 80ª palavra mais comum e três vezes mais que a 120ª palavra mais comum.

Uma distribuição de Zipf aparece na Figura 7.35(a). Ela captura a noção de que existe um pequeno número de itens populares e muitos itens não populares. Para reconhecer as distribuições dessa forma, é conveniente desenhar os dados em uma escala logarítmica nos dois eixos, como mostra a Figura 7.35(b). O resultado deverá ser uma linha reta.

Quando as pessoas viam a popularidade das páginas Web, isso também seguia aproximadamente a lei de Zipf (Breslay et al., 1999). Uma distribuição de Zipf é um exemplo em uma família de distribuições conhecidas como **leis da potenciação**. As leis da potenciação são evidentes em muitos fenômenos humanos, como a distribuição de populações e riquezas na cidade. Elas têm a mesma propensão para descrever poucos grandes participantes e muitos participantes menores, e também aparecem como uma linha reta em um gráfico logarítmico nos dois eixos. Logo, descobriu-se que a topologia da Internet poderia ser descrita de forma aproximada com leis da potenciação (Faloutsos et al., 1999). Em seguida, os pesquisadores começaram a desenhar cada propriedade imaginável da Internet em uma escala logarítmica, observando uma linha reta e gritando: 'Lei da potenciação!'

Entretanto, o que importa mais do que uma linha reta em um gráfico logarítmico nos dois eixos é o que essas distribuições significam para o projeto e uso de redes. Dadas as muitas formas de conteúdo que possuem distribuições da lei de Zipf ou da potenciação, parece fundamental que os sites na Internet sejam como Zipf em popularidade. Isso, por sua vez, significa que um site normal não é uma representação útil. Os sites são mais bem descritos como populares ou impopu-

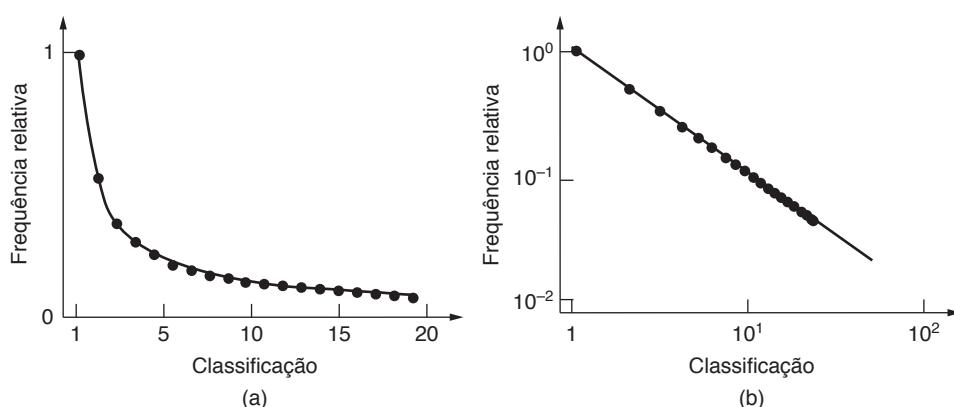



Figura 7.35 | Distribuição de Zipf (a) em uma escala linear e (b) em uma escala logarítmica nos dois eixos.

lares. Os dois tipos importam. A importância dos sites populares é óbvia, pois poucos deles já podem ser responsáveis pela maior parte do tráfego na Internet. Talvez seja surpresa saber que os sites impopulares também importam. Isso porque a quantidade total de tráfego direcionado para os sites impopulares pode se acumular e se tornar uma grande fração do tráfego geral. O motivo é que existem muitos sites impopulares. A noção de que, coletivamente, muitas escolhas impopulares podem importar tem sido propagada por livros como *The Long Tail* — A longa cauda (Anderson, 2008a).

Curvas que mostram decaimento, como a da Figura 7.35(a), são comuns, mas não são iguais. Em particular, situações em que a taxa de decaimento é proporcional a quanto material resta (como nos átomos radioativos instáveis) apresentam **decaimento exponencial**, que cai muito mais rapidamente do que a lei de Zipf. O número de itens, digamos, átomos, restantes após o tempo t normalmente é expresso como $e^{-t/\alpha}$, onde a constante α determina a rapidez do decaimento. A diferença entre o decaimento exponencial e a lei de Zipf é que, com o decaimento exponencial, é seguro ignorar o final da cauda, mas com a lei de Zipf o peso total da cauda é significativo e não pode ser ignorado.

Para trabalhar de modo eficaz nesse mundo tendencioso, precisamos ser capazes de construir os dois tipos de sites. Os impopulares são fáceis de lidar. Usando DNS, muitos sites diferentes podem realmente apontar para o mesmo computador na Internet que executa todos os sites. Por outro lado, sites populares são difíceis de lidar. Não existe um único computador, até mesmo remotamente poderoso o suficiente, e o uso de um único computador tornaria o site inacessível para milhões de usuários se ele falhasse. Para lidar com esses sites, temos que criar sistemas de distribuição de conteúdo. Vamos averiguar isso em seguida.

7.5.2 Parques de servidores e proxies Web

Os projetos da Web que vimos até aqui têm uma única máquina servidora falando com várias máquinas clientes. Para montar grandes sites que funcionem bem, podemos agilizar o processamento no lado do servidor ou no lado do cliente. No lado do servidor, podem ser montados servidores mais poderosos como um parque de servidores, em que um cluster ou conjunto de computadores atua como um único servidor. No lado do cliente, o melhor desempenho pode ser alcançado com melhores técnicas de caching. Em particular, os caches proxy oferecem um grande cache compartilhado para um grupo de clientes.

Vamos descrever cada uma dessas técnicas, uma por vez. No entanto, observe que nenhuma delas é suficiente para montar os maiores sites. Esses sites populares exigem os métodos de distribuição de conteúdo que descreveremos nas próximas seções, que combinam comunicações em muitos locais.

Parques de servidores

Não importa quanta largura de banda uma máquina tenha, ela só pode atender a muitas solicitações da Web enquanto a carga não for muito grande. A solução, nesse caso, é usar mais de um computador para criar um servidor Web. Isso nos leva ao modelo de **parque de servidores** da Figura 7.36.

A dificuldade com esse modelo aparentemente simples é que o conjunto de computadores que compõem o parque de servidores precisa se parecer com um único site lógico para os clientes. Se não, teremos simplesmente montado diferentes sites que funcionam em paralelo.

Existem várias soluções possíveis para fazer com que o conjunto de servidores pareça ser um único site. Todas as soluções assumem que qualquer um dos servidores pode lidar com uma solicitação de qualquer cliente. Para fazer isso, cada servidor precisa ter uma cópia do site. Os servidores aparecem como conectados a um banco de dados back-end comum por meio de uma linha tracejada para essa finalidade.

Uma solução é usar DNS para espalhar as solicitações pelos servidores no parque de servidores. Quando uma solicitação de DNS é feita para o URL da Web, o servidor DNS retorna uma lista móvel dos endereços IP dos servidores. Cada cliente experimenta um endereço IP, normalmente o primeiro na lista. O efeito é que diferentes clientes fazem contato com diferentes servidores para acessar o mesmo site, como era previsto. O método do DNS está no núcleo das CDNs, e retornaremos a ele mais adiante nesta seção.

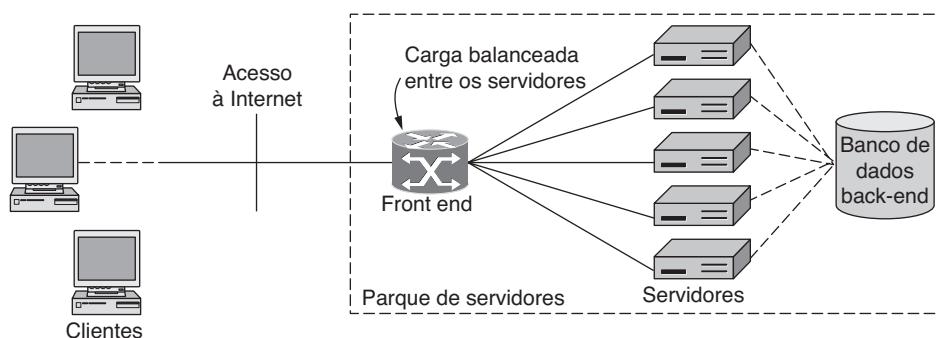



Figura 7.36 | Um parque de servidores.

As outras soluções são baseadas em um **front end** que espalha as solicitações que chegam pelo pool de servidores no parque de servidores. Isso acontece mesmo quando o cliente contata o parque de servidores usando um endereço IP de destino. O front end normalmente é um switch da camada de enlace ou um roteador IP, ou seja, um dispositivo que trata de quadros ou pacotes. Todas as soluções são baseadas no front end (ou nos servidores) lendo os cabeçalhos da camada de rede, transporte ou aplicação e usando-os de maneiras fora do padrão. Uma solicitação e uma resposta na Web são transportadas como uma conexão TCP. Para funcionar corretamente, o front end precisa distribuir todos os pacotes para uma solicitação ao mesmo servidor.

Um projeto simples é que o front end envie por broadcast, para todos os servidores, todas as solicitações que chegarem. Cada servidor responde somente a uma fração das solicitações por acordo anterior. Por exemplo, 16 servidores poderiam examinar o endereço IP de origem e responder ao solicitante somente se os 4 últimos bits do endereço IP de origem combinarem com seus seletores configurados. Outros pacotes são descartados. Embora isso seja um desperdício da largura de banda que chega, normalmente as respostas são muito maiores do que a solicitação, de modo que não é tão ineficaz quanto possa parecer.

Em um projeto mais geral, o front end pode inspecionar os cabeçalhos IP, TCP e HTTP dos pacotes e mapeá-los arbitrariamente para um servidor. O mapeamento é chamado política de **balanceamento de carga**, pois o objetivo é balancear a carga de trabalho entre os servidores. A política pode ser simples ou complexa. Uma política simples poderia usar os servidores um após o outro, um por vez, ou em forma de rodízio. Com essa técnica, o front end precisa se lembrar do mapeamento para cada solicitação, de modo que os pacotes seguintes que fazem parte da mesma solicitação sejam enviados ao mesmo servidor. Além disso, para tornar o site mais confiável do que um único servidor, o front end deve observar quando os servidores falharam e parar de enviar solicitações a eles.

Assim como o NAT, esse projeto geral é arriscado, ou pelo menos frágil, porque acabamos de criar um dispositivo que viola o princípio mais básico dos protocolos em camadas: cada camada precisa usar seu próprio cabeçalho para fins de controle e não pode inspecionar e usar informações da carga útil para qualquer finalidade. Mas as pessoas projetam sistemas desse tipo de qualquer forma, e quando eles deixam de funcionar no futuro, devido a mudanças nas camadas mais altas, elas costumam ficar surpresas. O front end nesse caso é um switch ou roteador, mas pode tomar uma ação com base na informação da camada de transporte ou outra mais alta. Essa caixa é chamada **middlebox**, pois se interpõe no meio de um caminho da rede em que não tem responsabilidade, de acordo com a pilha de protocolos. Nesse caso, o front end é considerado uma parte interna de um parque de servidores, que termina todas as camadas até a camada de aplicação (e por isso pode usar toda a informação do cabeçalho para essas camadas).

Apesar disso, assim como NAT, esse projeto é útil na prática. O motivo para examinar cabeçalhos TCP é que é possível realizar um trabalho melhor de balanceamento de carga do que apenas com a informação do IP. Por exemplo, um endereço IP pode representar uma empresa inteira e fazer muitas solicitações. É apenas examinando o TCP ou informações de camada mais alta que essas solicitações podem ser mapeadas para diferentes servidores.

O motivo para examinar os cabeçalhos HTTP é um pouco diferente. Muitas interações Web acessam e atualizam bancos de dados, como quando um cliente examina sua compra mais recente. O servidor que atende essa solicitação terá que consultar o banco de dados back-end. É útil direcionar as próximas solicitações do mesmo usuário para o mesmo servidor, pois esse servidor já tem informações em cache sobre o usuário. O modo mais simples de fazer com que isso aconteça é usar cookies da Web (ou outra informação para distinguir o usuário) e inspecionar os cabeçalhos HTTP para encontrar os cookies.

Como uma nota final, embora tenhamos descrito esse projeto para sites, um parque de servidores também pode ser montado para outros tipos de servidores. Um exemplo são os servidores de streaming de mídia por UDP. A única mudança necessária é que o front end seja capaz de balancear a carga dessas solicitações (que terá campos no cabeçalho do protocolo diferentes daqueles das solicitações Web).

Proxies Web

Solicitações e respostas da Web são enviadas usando HTTP. Na Seção 7.3, descrevemos como os navegadores podem manter respostas em cache e reutilizá-las para responder a solicitações futuras. Vários campos de cabeçalho e regras são usados pelo navegador para determinar se uma cópia em cache de uma página Web ainda é válida. Não repetiremos esse material aqui.

O caching melhora o desempenho encurtando o tempo de resposta e reduzindo a carga na rede. Se o navegador puder determinar que uma página em cache é válida por si só, a página poderá ser capturada do cache imediatamente, sem nenhum tráfego na rede. Porém, mesmo que o navegador tenha que pedir confirmação do servidor para saber se a página é válida, o tempo de resposta é curto e a carga da rede é reduzida, especialmente para páginas grandes, pois apenas uma mensagem pequena precisa ser enviada.

Porém, o melhor que o navegador pode fazer é manter em cache todas as páginas que o usuário visitou anteriormente. Pela nossa discussão sobre popularidade, você deve se lembrar que, assim como algumas páginas populares que muitas pessoas visitam repetidamente, existem muitas e muitas páginas impopulares. Na prática, isso limita a eficácia do caching do navegador, pois há um grande número de páginas que são visitadas apenas uma vez por determinado usuário. Essas páginas sempre precisam ser capturadas do servidor.

Uma estratégia para tornar os caches mais eficazes é compartilhá-los entre vários usuários. Desse modo, uma página já capturada para um usuário pode ser retornada para outro quando este fizer a mesma solicitação. Sem o caching do navegador, os dois usuários precisariam buscar a página do servidor. É claro que esse compartilhamento não pode ser feito para o tráfego criptografado, páginas que exigem autenticação e páginas que não podem ser mantidas em cache (por exemplo, preços atuais de ações), que são retornadas por programas. As páginas dinâmicas criadas por programas, especialmente, são um caso crescente para o qual o caching não é eficaz. Apesar disso, existem muitas páginas que são visíveis a muitos usuários e se parecem iguais, não importa qual usuário faça a solicitação (por exemplo, imagens).

Um **proxy Web** é usado para compartilhar cache entre os usuários. Um proxy é um agente que atua em favor de alguém, como o usuário. Existem muitos tipos de proxies. Por exemplo, um proxy ARP responde a solicitações ARP em favor de um usuário que está em outro lugar (e não pode responder por si mesmo). Um proxy Web busca solicitações Web em favor de seus usuários. Ele normalmente oferece caching das respostas da Web e, por ser compartilhado pelos usuários, tem cache substancialmente maior do que um navegador.

Quando um proxy é usado, a configuração típica é que uma organização opere um proxy Web para todos os seus usuários. A organização pode ser uma empresa ou um ISP. Ambos buscam benefícios agilizando as solicitações Web para seus usuários e reduzindo suas necessidades de largura de banda. Embora um preço fixo, independentemente do uso, seja comum para os usuários finais, a maioria das empresas e ISPs é cobrada de acordo com a largura de banda que elas utilizam.

Essa configuração aparece na Figura 7.37. Para usar o proxy, cada navegador é configurado para fazer solicitações de página ao proxy, e não ao servidor real da página. Se o proxy tiver a página, ele a retornará imediatamente. Se não, ele buscará a página do servidor e a acrescentará em cache para uso futuro, depois disso, ele a retornará ao cliente que a solicitou.

Assim como o envio de solicitações da Web ao proxy em vez de ao servidor real, os clientes realizam seu próprio caching usando o cache do seu navegador. O proxy só é consultado depois que o navegador tiver tentado satisfazer a solicitação por seu próprio cache. Ou seja, o proxy oferece um segundo nível de caching.

Outros proxies podem ser acrescentados, para oferecer níveis adicionais de caching. Cada proxy (ou o navegador) faz solicitações por meio de seu **proxy upstream**. Cada proxy upstream mantém caches para os **proxies downstream** (ou navegadores). Assim, é possível que os navegadores em uma empresa usem um proxy da empresa, que usa um proxy do ISP, que entra em contato com os servidores Web diretamente. Porém, o único nível de caching de proxy que mostramos na Figura 7.37 normalmente é suficiente para obter o máximo dos benefícios em potencial, na prática. O problema novamente é a longa cauda da popularidade. Os estudos do tráfego da Web mostraram que o caching compartilhado é especialmente benéfico até que o número de usuários alcance aproximadamente o tamanho de uma pequena empresa (digamos, 100 pessoas). Quando o número de pessoas cresce muito, os benefícios do compartilhamento de um cache se tornam menores, devido às solicitações impopulares que não podem ser mantidas em cache por causa da falta de espaço de armazenamento (Wolman et al., 1999).

Os proxies Web oferecem benefícios adicionais que normalmente são um fator importante na decisão sobre seu emprego. Um benefício é filtrar o conteúdo. O administrador pode configurar o proxy para impedir sites ou filtrar de alguma outra maneira as solicitações que ele faz. Por exemplo, muitos administradores não gostam que os funcionários assistam aos vídeos do YouTube (ou, pior ainda, pornografia) no horário de trabalho, e definem seus filtros de forma a impedi-los. Outro benefício de ter proxies é a privacidade e o anonimato, quando o proxy encobre do servidor a identidade do usuário.

7.5.3 Redes de entrega de conteúdo

Parques de servidores e proxies Web ajudam a criar grandes sites e melhorar o desempenho da Web, mas eles não são suficientes para sites verdadeiramente populares, que precisam enviar conteúdo em escala global. Para esses sites, uma técnica diferente é necessária.

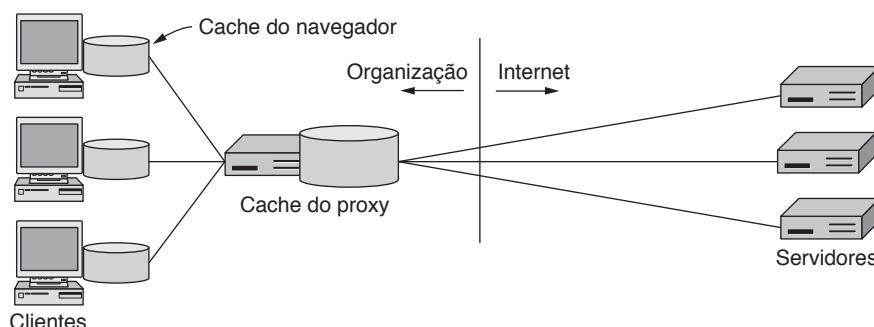



Figura 7.37 | Cache de um proxy entre navegadores e servidores Web.

As redes de entrega de conteúdo, ou **CDNs** (**Content Delivery Networks**), viram a ideia do caching da Web tradicional de ponta-cabeça. Em vez de fazer com que os clientes procurem uma cópia da página solicitada em um cache próximo, é o provedor que coloca uma cópia da página em um conjunto de nós em diferentes locais e instrui o cliente a usar um nó vizinho como servidor.

Um exemplo do caminho que os dados seguem quando são distribuídos por uma CDN pode ser visto na Figura 7.38. O servidor de origem na CDN distribui uma cópia do conteúdo para outros nós na CDN, em Sidney, Boston e Amsterdã, neste exemplo. Isso é mostrado com linhas tracejadas. Os clientes então buscam páginas do nó mais próximo na CDN, o que é mostrado com as linhas sólidas. Desse modo, os dois clientes em Sidney buscam a cópia da página que está armazenada em Sidney; nenhum deles busca a página do servidor de origem, que pode estar na Europa.

O uso de uma estrutura em árvore tem três vantagens. Primeiro, a distribuição de conteúdo pode ser expandida para tantos clientes quantos forem necessários, usando mais nós na CDN e mais níveis na árvore, quando a distribuição entre os nós das CDNs se tornar o gargalo. Não importa quantos clientes existam, a estrutura em árvore é eficiente. O servidor de origem não fica sobrecarregado, pois ele fala com os muitos clientes por meio da árvore de nós da CDN; ele mesmo não precisa responder a cada solicitação de página. Segundo, cada cliente recebe um bom desempenho, buscando páginas de um servidor próximo ao invés das de um servidor distante. Isso porque o tempo de ida e volta para estabelecer uma conexão é mais curto, a partida lenta do TCP sobe mais rapidamente devido ao tempo de ida e volta mais curto e o caminho de rede mais curto terá menos chance de passar por regiões de congestionamento na Internet. Finalmente, a carga total sobre a rede também é mantida no mínimo. Se os nós das CDNs forem bem posicionados, o tráfego para determinada página deverá passar por cada parte da rede apenas uma vez. Isso é importante porque, no fim, alguém está pagando pela largura de banda da rede.

A ideia do uso de uma árvore de distribuição é simples. O que é mais complicado é como organizar os clientes para usar essa árvore. Por exemplo, os servidores proxy parecem oferecer uma solução. Examinando a Figura 7.38, se cada cliente foi configurado para usar o nó CDN de Sidney, Boston ou Amsterdã como um proxy Web de caching, a distribuição seguiria a árvore. Porém, essa estratégia não é suficiente na prática, por três motivos. O primeiro motivo é que os clientes em determinada parte da rede provavelmente pertencem a diferentes organizações, de modo que provavelmente estão usando proxies Web diferentes. Lembre-se de que caches normalmente não são compartilhados entre as organizações, devido ao benefício limitado do caching a um grande número de clientes, e também por questões de segurança. Segundo, pode haver várias CDNs, mas cada cliente só usa um único cache proxy. Qual CDN um cliente deverá usar como seu proxy? Finalmente, talvez o problema mais prático de todos seja que os proxies Web são configurados pelos clientes. Eles podem ou não ser configurados para que se beneficiem da distribuição de conteúdo por uma CDN, e há muito pouco que a CDN pode fazer quanto a isso.

Outro modo simples de dar suporte à árvore de distribuição de um nível é usar a técnica do **espelhamento**. Nessa técnica, o servidor de origem replica o conteúdo pelos nós da CDN como antes. Os nós da CDN em diferentes regiões da rede são chamados de **espelhos**. As páginas Web no servidor de origem contêm links explícitos para os diferentes espelhos, normalmente informando seu local ao usuário. Esse projeto permite que o usuário selecione manualmente um espelho próximo para usar para baixar conteúdo. Um uso típico do espelhamento é colocar um grande pacote de software em espelhos localizados, por exemplo, nas costas Leste e Oeste dos Estados Unidos, Ásia e Europa. Os sites espelhados geralmente são completamente estáticos, e a escolha dos sites permanece sendo

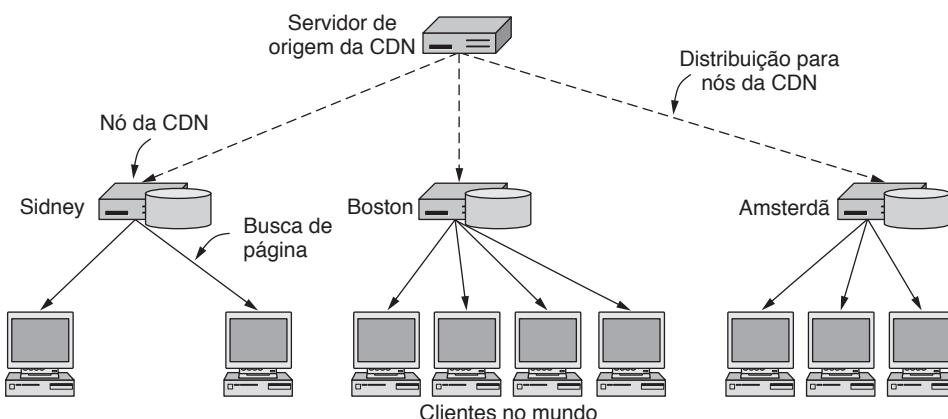



Figura 7.38 | Árvore de distribuição da CDN.

estável por meses ou anos. Eles compõem uma técnica testada e aprovada. Porém, dependem do usuário para fazer a distribuição, pois os espelhos são realmente sites Web diferentes, mesmo que estejam ligados.

A terceira técnica, que contorna as dificuldades das duas técnicas anteriores, utiliza DNS e é chamada **redirecionamento de DNS**. Suponha que um cliente queira buscar uma página com o URL http://www.cdn.com/page.html. Para buscar a página, o navegador usará o DNS para transformar www.cdn.com em um endereço IP. Essa pesquisa de DNS prossegue normalmente. Usando o protocolo DNS, o navegador descobre o endereço IP do servidor de nomes para cdn.com, depois entra em contato com o servidor de nomes para lhe pedir que resolva www.cdn.com. Agora vem a parte realmente inteligente. O servidor de nomes é executado pela CDN. Em vez de retornar o mesmo endereço IP para cada solicitação, ele pesquisará o endereço IP do cliente que faz a solicitação e retornará respostas diferentes. A resposta será o endereço IP do nó da CDN que está mais próximo do cliente. Ou seja, se um cliente em Sidney pedir ao servidor de nomes da CDN para resolver www.cdn.com, o servidor de nomes retornará o endereço IP do nó da CDN de Sidney, mas se um cliente em Amsterdã fizer a mesma solicitação, o servidor de nomes retornará o endereço IP do nó da CDN em Amsterdã em seu lugar.

Essa estratégia é perfeitamente válida, de acordo com a semântica do DNS. Já vimos que os servidores de nomes podem retornar listas mutáveis de endereços IP. Depois da resolução de nomes, o cliente em Sidney buscará a página diretamente do nó da CDN de Sidney. Outras páginas no mesmo 'servidor' também serão capturadas diretamente do nó da CDN de Sidney, graças ao caching DNS. A sequência geral das etapas aparece na Figura 7.39.

Uma questão complexa nesse processo é o que significa encontrar o nó da CDN mais próximo e como tratar disso. Para definir o mais próximo, não é a geometria que importa. Existem pelo menos dois fatores a considerar no mapeamento de um cliente para um nó da CDN. Um fator é a distância da rede. O cliente deverá ter um caminho de rede curto e de alta capacidade para o nó da CDN. Essa situação produzirá downloads rápidos. As CDNs usam um mapa que elas calcularam anteriormente para traduzir entre o endereço IP de um cliente e seu local na rede. O nó da CDN que é selecionado poderia ser aquele na distância mais curta em uma linha reta, ou não. O que importa é alguma combinação do tamanho do caminho da rede com quaisquer limites de capacidade ao longo dela. O segundo fator é a carga que já está sendo transportada pelo nó da CDN. Se os nós da CDN estiverem sobrecarregados, eles entregarão respostas lentamente, assim como o servidor Web sobrecarregado que buscamos evitar em primeiro lugar. Assim, pode ser preciso balancear a carga entre os nós das CDN, mapeando alguns clientes para nós que estão um pouco mais afastados, porém menos sobrecarregados.

As técnicas para usar o DNS para distribuição de conteúdo foram iniciadas pela Akamai a partir de 1998, quando a Web estava vergando sob a carga de seu crescimento inicial. A Akamai foi a primeira CDN importante, e tornou-se líder na indústria. Provavelmente ainda mais inteligente do que a ideia de usar DNS para conectar clientes a nós vizinhos tenha sido a estrutura de incentivo dos negócios deles. As empresas pagam à Akamai para oferecer seu conteúdo aos clientes, de modo que elas tenham sites Web responsivos que os clientes gostem de usar. Os nós CDN precisam ser colocados em locais da rede com boa conectividade, o que inicialmente significava dentro das redes do ISP. Para os ISPs, existe um benefício em ter um nó CDN em suas redes, já que o nó da CDN reduz a quantidade de largura de banda de upstream da rede de que elas precisam (e pela qual devem pagar), assim como os caches proxy. Além disso, o nó CDN melhora a responsividade para os clientes do ISP, fazendo com que se pareça melhor aos seus olhos, dando-lhes uma vantagem competitiva sobre os ISPs que não possuem um nó CDN. Esses benefícios (sem custo para o ISP) tornam a instalação de um nó CDN uma escolha óbvia para o ISP. Assim, o provedor de conteúdo, o ISP e os clientes se beneficiam, e a CDN ganha dinheiro. Desde 1998, outras empresas entraram no negócio, de modo que esse agora é um ramo competitivo, com vários provedores.

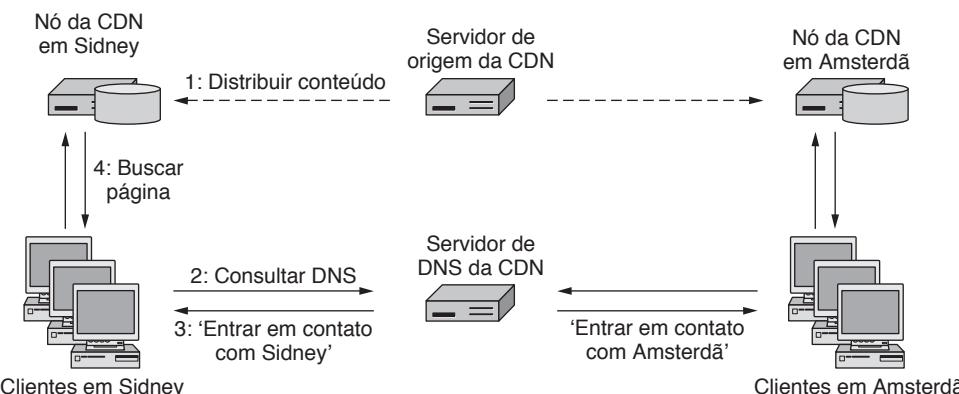



Figura 7.39 | Direcionando clientes para nós CDN vizinhos usando DNS.

Como essa descrição indica, a maioria das empresas não monta sua própria CDN. Em vez disso, elas usam os serviços de um provedor de CDN, como a Akamai, para realmente fornecer seu conteúdo. Para permitir que outras empresas usem o serviço de uma CDN, precisamos acrescentar uma última etapa à nossa figura.

Depois que o contrato for assinado para uma CDN distribuir o conteúdo em favor do proprietário de um site Web, o proprietário entrega o conteúdo à CDN. Esse conteúdo é enviado aos nós da CDN. Além disso, o proprietário reescreve quaisquer páginas Web que se vinculam ao conteúdo. Em vez de se vincularem ao conteúdo do seu site, as páginas se vinculam ao conteúdo por meio da CDN. Como exemplo de funcionamento desse esquema, considere o código fonte para a página Web da Fluffy Video, que aparece no Quadro 7.12(a). Após o pré-processamento, ele é transformado para o Quadro 7.12(b) e colocado no servidor da Fluffy Video como www.fluffyvideo.com/index.html.

Quando um usuário digita o URL www.fluffyvideo.com em seu navegador, o DNS retorna o endereço IP do próprio site da Fluffy Video, permitindo que a página principal (HTML) seja capturada normalmente. Quando o usuário clica em qualquer um dos hiperlinks, o navegador pede ao DNS para procurar www.cdn.com. Essa pesquisa faz contato com o servidor DNS da CDN, que retorna o endereço IP do nó da CDN mais próximo. O navegador, então, envia uma solicitação HTTP normal para o nó da CDN, por exemplo, /fluffyvideo/koalas.mpg. O URL identifica a página a retornar, começando o caminho com *fluffyvideo*, para que o nó da CDN possa separar as solicitações para as diferentes empresas a que atende. Por fim, o vídeo é retornado e o usuário pode ver os animais de pelúcia.

A estratégia por trás dessa divisão de conteúdo hospedado pela CDN e páginas de entrada hospedadas pelo proprietário do conteúdo é que ela dá o controle ao proprietário do conteúdo, enquanto permite que a CDN mova a massa de dados. A maioria das páginas de entrada é pequena, sendo apenas texto HTML. Essas páginas costumam ser vinculadas a grandes arquivos, como vídeos e imagens. São de fato esses grandes arquivos que são atendidos pela CDN, embora o uso de uma CDN seja completamente transparente aos usuários. O site se parece o mesmo, mas funciona de modo mais rápido.

Existe outra vantagem para os sites que usam uma CDN compartilhada. A demanda futura para um site pode ser difícil de prever. Com frequência, existem surtos de

```
<html>
<head> <title> Fluffy Video </title> </head>
<body>
<h1> Fluffy Video's Product List </h1>
<p> Click below for free samples. </p>

<a href = "koalas.mpg"> Koalas Today </a> <br>
<a href = "kangaroos.mpg"> Funny Kangaroos </a> <br>
<a href = "wombats.mpg"> Nice Wombats </a> <br>
</body>
</html>
```

(a)

```
<html>
<head> <title> Fluffy Video </title> </head>
<body>
<h1> Fluffy Video's Product List </h1>
<p> Click below for free samples. </p>

<a href = "http://www.cdn.com/fluffyvideo/koalas.mpg"> Koalas Today </a> <br>
<a href = "http://www.cdn.com/fluffyvideo/kangaroos.mpg"> Funny Kangaroos </a> <br>
<a href = "http://www.cdn.com/fluffyvideo/wombats.mpg"> Nice Wombats </a> <br>
</body>
</html>
```

(b)

Quadro 7.12 | (a) Página Web original. (b) A mesma página após vincular à CDN.

demanda, conhecidos como **flash crowds**. Tais surtos podem acontecer quando o produto mais recente é lançado, quando há um desfile de modas ou outro evento, ou então quando a empresa está em destaque de alguma outra maneira. Até mesmo um site que era desconhecido, pouco visitado, de repente pode se tornar o foco da Internet se for merecedor e estiver vinculado a sites populares. Como a maioria dos sites não está preparada para lidar com aumentos maciços no tráfego, o resultado é que muitos deles falham quando há um surto de tráfego.

Veja um caso. De modo geral, o site da Secretaria do Estado da Flórida não é um local muito ocupado, embora você possa pesquisar nele informações sobre empresas da Flórida, cartórios e assuntos culturais, além de informações sobre votação e eleições. Por alguma estranha razão, em 7 de novembro de 2000 (a data da eleição presidencial nos Estados Unidos, com Bush contra Gore), muitas pessoas repentinamente se interessaram pela página de resultados da eleição desse site. O site de repente se tornou um dos mais ocupados do mundo e, naturalmente, falhou em consequência disso. Se ele estivesse em uma CDN, provavelmente teria sobrevivido.

Usando uma CDN, um site tem acesso a uma capacidade de atendimento de conteúdo muito grande. As maiores CDNs possuem dezenas de milhares de servidores implantados em países do mundo inteiro. Como apenas um pequeno número de sites estará experimentando um flash crowd em determinado momento (por definição), esses sites podem usar a capacidade da CDN para lidar com a carga até que a tempestade passe. Ou seja, a CDN pode rapidamente aumentar a capacidade de atendimento de um site.

Essa discussão é uma descrição simplificada de como a Akamai funciona. Existem muitos outros detalhes que importam na prática. Os nós CDN representados em nosso exemplo normalmente são clusters de máquinas. O redirecionamento de DNS é feito em dois níveis: um para mapear clientes para o local aproximado da rede e outro para espalhar a carga sobre os nós nesse local. Alguns aspectos envolvidos são confiabilidade e desempenho. Para poder deslocar um cliente de uma máquina em um cluster para outra, as respostas do DNS no segundo nível são dadas com TTLs curtos, de modo que o cliente repetirá a resolução após um curto período. Finalmente, embora tenhamos nos concentrado nos aspectos de distribuição de objetos estáticos, como imagens e vídeos, as CDNs também podem aceitar a criação de página dinâmica, streaming de mídia e outros. Para obter mais informações sobre CDNs, consulte Dilley et al. (2002).

7.5.4 Redes peer-to-peer

Nem todos podem montar uma CDN de mil nós em locais no mundo inteiro para distribuir seu conteúdo. (Na verdade, não é difícil alugar mil máquinas virtuais no mundo inteiro, devido à indústria de hospedagem bem desenvolvida e altamente competitiva. Porém, conseguir os nós é só o início da montagem de uma CDN.) Felizmente, existe uma alternativa para o restante de nós, que é simples de usar e pode distribuir uma quantidade tremenda de conteúdo. É uma rede P2P (Peer-to-Peer).

As redes P2P entraram em cena a partir de 1999. A primeira aplicação divulgada foi para o crime em massa: 50 milhões de usuários do Napster estavam trocando músicas sem permissão dos proprietários do direito autoral, até que a Napster foi fechada pelos tribunais, em meio a uma grande controvérsia. Apesar disso, a tecnologia peer-to-peer tem muitos usos interessantes e legais. Outros sistemas continuaram em desenvolvimento, com tão grande interesse dos usuários que o tráfego P2P rapidamente encobriu o tráfego da Web. Hoje, BitTorrent é o protocolo P2P mais popular. Ele é muito usado para compartilhar vídeos (licenciados e de domínio público) e outros conteúdos, sendo responsável por uma grande fração de todo o tráfego na Internet. Vamos examiná-lo nesta seção.

A ideia básica de uma rede de compartilhamento de arquivos **P2P (Peer-to-Peer)** é que muitos computadores se juntam e compartilham seus recursos para formar um sistema de distribuição de conteúdo. Os computadores normalmente são apenas computadores domésticos. Eles não precisam ser máquinas nos centros de dados da Internet. Os computadores são chamados peers (pares) porque cada um pode alternadamente atuar como um cliente para outro peer, buscando seu conteúdo, e como servidor, fornecendo conteúdo para outros peers. O que torna os sistemas peer-to-peer interessantes é que não existe uma infraestrutura dedicada, diferente de uma CDN. Qualquer um participa na tarefa de distribuir conteúdo e normalmente não existe um ponto de controle central.

Muitas pessoas estão entusiasmadas com a tecnologia P2P porque ela é vista como algo que dá poder aos pequenos. O motivo não é apenas que é necessária uma grande empresa para conduzir uma CDN, enquanto qualquer um com um computador pode se juntar a uma rede P2P. É que as redes P2P possuem uma formidável capacidade de distribuir conteúdo que pode se igualar ao maior dos sites Web.

Considere uma rede P2P composta de N usuários comuns, cada um com uma conectividade de banda larga a 1 Mbps. A capacidade de upload agregada da rede P2P, ou taxa em que os usuários podem enviar tráfego para a Internet, é de N Mbps. A capacidade de download, ou a taxa em que os usuários podem receber tráfego, também é N Mbps. Cada usuário também pode fazer upload ou download ao mesmo tempo, pois tem um enlace de 1 Mbps em cada direção.

Não é óbvio que isso seja verdadeiro, mas acontece que toda a capacidade pode ser usada produtivamente para distribuir conteúdo, mesmo para o caso de compartilhamento de uma única cópia de um arquivo com todos os outros usuários. Para ver como isso pode acontecer, imagine que

os usuários estejam organizados em uma árvore binária, com cada usuário não-folha enviando para dois outros usuários. A árvore transportará a única cópia do arquivo para todos os outros usuários. Para usar a largura de banda de upload para o máximo de usuários possível o tempo todo (e, portanto, distribuir um arquivo grande com baixa latência), precisamos canalizar a atividade de rede dos usuários. Imagine que o arquivo esteja dividido em mil partes. Cada usuário pode receber uma nova parte de algum lugar acima na árvore e enviar a parte previamente recebida para baixo na árvore ao mesmo tempo. Desse modo, quando o pipeline for iniciado, depois que um pequeno número de partes (igual à profundidade da árvore) for enviado, todos os usuários não-folhas estarão ocupados fazendo o upload do arquivo para outros usuários. Como existem aproximadamente $N/2$ usuários não-folhas, a largura de banda de upload dessa árvore é $N/2$ Mbps. Podemos repetir esse truque e criar outra árvore que usa os outros $N/2$ Mbps de largura de banda de upload trocando os papéis entre os nós folhas e os nós que não são folhas. Essa construção conjunta usa toda a capacidade.

Esse argumento significa que as redes P2P são autoescaláveis. Sua capacidade de upload útil cresce em conjunto com as demandas de download que podem ser feitas por seus usuários. Elas sempre são 'grandes o suficiente' em certo sentido, sem a necessidade de alguma infraestrutura dedicada. Ao contrário, a capacidade até mesmo de um site Web grande é fixa e será ou muito grande ou muito pequena. Considere um site com apenas 100 clusters, cada um capaz de trabalhar a 10 Gbps. Essa capacidade enorme não ajuda quando existe um pequeno número de usuários. O site não pode receber informações de N usuários em uma taxa mais rápida que N Mbps, pois o limite está nos usuários, e não no site. E quando existe mais de um milhão de usuários a 1 Mbps, o site Web não pode bombear dados com rapidez suficiente para manter todos eles ocupados fazendo download. Pode parecer que esse é um grande número de usuários, mas grandes redes BitTorrent (por exemplo, Pirate Bay) afirmam ter mais de 10 milhões de usuários. Isso significa algo como 10 terabits/s em termos do nosso exemplo.

Você deverá considerar esses números com alguma cautela, pois são uma simplificação da situação real. Um desafio significativo para redes P2P é usar bem a largura de banda quando os usuários chegarem de todas as formas e tamanhos, e tiverem diferentes capacidades de download e upload. Apesar disso, esses números indicam o enorme potencial do P2P.

Existe outro motivo para as redes P2P serem importantes. CDNs e outros serviços executados de forma centralizada colocam os provedores em posição de ter uma série de informações pessoais sobre muitos usuários, desde preferências de navegação e locais onde as pessoas compram on-line até localizações e endereços de e-mail. Essas informações podem ser usadas para oferecer um serviço melhor e mais personalizado, ou para a invasão da privacidade das pessoas. Este último pode acontecer intencionalmente — digamos, como parte de um novo produto — ou através de divulgação ou comprometimento acidental. Com os sistemas P2P, não há um único provedor capaz de monitorar o sistema inteiro. Isso não significa que os sistemas P2P necessariamente oferecerão privacidade, pois os usuários estão confiando uns nos outros até certo ponto. Isso só significa que eles podem oferecer uma forma diferente de privacidade do que sistemas administrados de forma centralizada. Os sistemas P2P agora estão sendo explorados para serviços além do compartilhamento de arquivos (por exemplo, armazenamento, streaming) e o tempo dirá se essa vantagem é significativa.

A tecnologia P2P tem seguido dois caminhos relacionados enquanto está sendo desenvolvida. No lado mais prático, existem sistemas usados todos os dias. Os mais conhecidos deles são baseados no protocolo BitTorrent. No lado mais acadêmico, tem havido um intenso interesse nos algoritmos DHT (Distributed Hash Table), que permitem aos sistemas P2P atuar como um todo, embora não contem com nenhum componentes centralizados. Vamos examinar essas duas tecnologias.

BitTorrent

O protocolo BitTorrent foi desenvolvido por Brahm Cohen em 2001 para permitir que um conjunto de peers compartilhasse arquivos rápida e facilmente. Existem dezenas de clientes disponíveis gratuitamente que entendem esse protocolo, assim como muitos navegadores que entendem o protocolo HTTP para servidores Web. O protocolo está disponível como um padrão aberto em www.bittorrent.org.

Em um sistema peer-to-peer típico, como aquele formado com o BitTorrent, cada um dos usuários tem a mesma informação que pode ser de interesse para outros usuários. Essa informação pode ser software gratuito, música, vídeos, fotografias etc. Existem três problemas que precisam ser resolvidos para compartilhar o conteúdo nesse ambiente:

1. como um peer encontra outros peers que possuem o conteúdo que ele deseja baixar?
2. como o conteúdo é replicado pelos peers para fornecer downloads de alta velocidade para qualquer um?
3. como os peers encorajam uns aos outros para que façam upload do conteúdo para outros, além do download de conteúdo para eles próprios?

O primeiro problema existe porque nem todos os peers terão todo o conteúdo, pelo menos inicialmente. A técnica usada no BitTorrent é que cada provedor de conteúdo possa criar uma descrição de conteúdo chamada **torrent**. A torrent é muito menor que o conteúdo, e é usada por um peer para verificar a integridade dos dados que ele baixa de outros peers. Outros usuários que querem baixar o conteúdo precisam primeiro obter a torrent, digamos, encontrando-a em uma página Web que anuncia o conteúdo.

A torrent é apenas um arquivo em um formato especificado, que contém dois tipos de informação. Um tipo é o nome de um **tracker**, que é um servidor que leva os peers ao conteúdo da torrent. O outro tipo de informação é uma lista de partes de mesmo tamanho, ou **chunks**, que compõem o conteúdo. Diferentes tamanhos de chunk podem ser usados para diferentes torrents, normalmente de 64 KB a 512 KB. O arquivo de torrent contém o nome de cada chunk, dado como um hash SHA-1 de 160 bits do chunk. Veremos os hashes criptográficos como SHA-1 no Capítulo 8. Por enquanto, você pode pensar em um hash como um checksum maior e mais seguro. Dado o tamanho dos chunks e dos hashes, o arquivo de torrent é pelo menos três ordens de grandeza menor que o conteúdo, de modo que pode ser transferido rapidamente.

Para baixar o conteúdo descrito em uma torrent, um peer primeiro entra em contato com o tracker para a torrent. O **tracker** é um servidor que mantém uma lista de todos os outros peers que estão ativamente fazendo download e upload do conteúdo. Esse conjunto de peers é chamado **swarm**. Os membros do swarm entram em contato com o tracker regularmente para informar que ainda estão ativos, bem como ao saírem do swarm. Quando um novo peer entra em contato com o tracker para se juntar ao swarm, o tracker lhe informa sobre outros peers no swarm. Capturar uma torrent e entrar em contato com o tracker são os dois primeiros passos para baixar conteúdo, como mostra a Figura 7.40.

O segundo problema é como compartilhar conteúdo de um modo que os downloads sejam rápidos. Quando um swarm é formado inicialmente, alguns peers precisam ter todos os chunks que compõem o conteúdo. Esses peers são chamados **seeders**. Outros peers que se juntam ao swarm não terão chunks; eles são os peers que estão baixando o conteúdo.

Enquanto um peer participa de um swarm, ele simultaneamente efetua o download de chunks que não encontra em outros peers e faz o upload de chunks dos quais outros peers precisam. Essa negociação é mostrada como a última etapa da distribuição de conteúdo na Figura 7.40. Com o tempo, o peer reúne mais chunks até que tenha baixado todo o conteúdo. O peer pode sair do swarm (e retornar) a qualquer momento. Normalmente, um peer permanecerá por um pequeno período depois que terminar seu próprio download. Com peers indo e vindo, a taxa de agitação em um swarm pode ser muito alta.

Para que o método anteriormente citado funcione bem, cada chunk deve estar disponível em muitos peers. Se cada um capturasse os chunks na mesma ordem, é provável que muitos peers dependessem dos seeders para o próximo chunk. Isso criaria um gargalo. Em vez disso, os peers trocam as listas dos chunks que possuem uns com os outros. Depois, eles selecionam chunks raros que são difíceis de encontrar para baixar. A ideia é que o download de um chunk raro crie uma cópia dele, o que tornará o chunk mais fácil para outros peers encontrarem e baixarem. Se todos os peers fizerem isso, depois de algum tempo, todos os chunks estarão disponíveis de modo generalizado.

O terceiro problema talvez seja o mais interessante. Os nós da CDN são preparados exclusivamente para fornecer conteúdo aos usuários. Os nós P2P não são. Eles são os computadores dos usuários, e os usuários podem estar mais interessados em conseguir um filme do que em ajudar outros usuários com seus downloads. Os nós que capturam recursos de um sistema sem contribuir em troca são chamados **free-riders** ou **leechers**. Se houver muitos deles, o sistema não funcionará bem. Os primeiros sistemas P2P eram conhecidos por hospedá-los (Saroiu et al., 2003), de modo que o BitTorrent procurou minimizá-los.

A técnica usada nos clientes BitTorrent é recompensar os peers que mostram bom comportamento de upload. Cada peer seleciona aleatoriamente os outros, capturando chunks deles enquanto faz o upload de chunks para eles. O peer continua a trocar chunks apenas com um pequeno número de peers que oferecem o desempenho de download mais alto, embora também experimentem aleatoria-

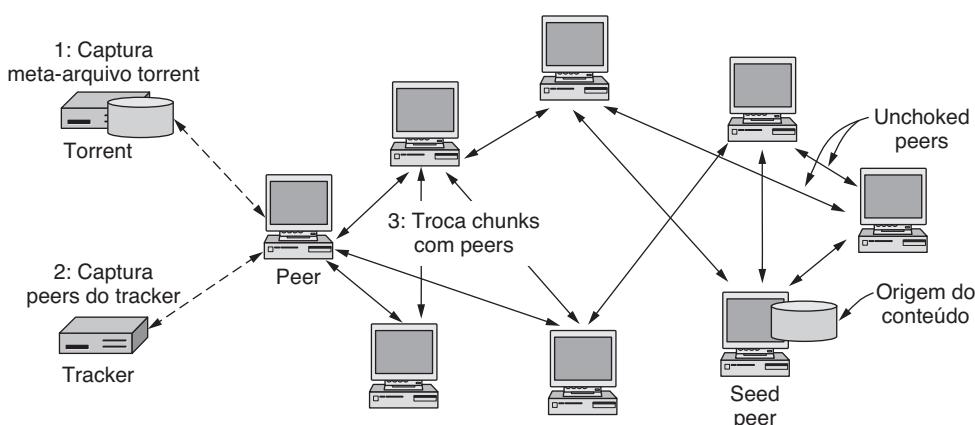



Figura 7.40 | BitTorrent.

mente outros peers para encontrar bons parceiros. Experimentar os peers aleatoriamente também permite que os novatos obtenham chunks iniciais que eles podem trocar com outros peers. Os peers com os quais um nó está trocando chunks atualmente são considerados **unchoked**.

Com o tempo, esse algoritmo deverá combinar peers com upload comparável e taxas de download uns com os outros. Quanto mais um peer estiver contribuindo com outros, mais ele poderá esperar em retorno. Usar um conjunto de peers também ajuda a saturar a largura de banda de download de um peer para aumentar o desempenho. Ao contrário, se um peer não estiver fazendo upload de chunks para outros peers, ou se estiver fazendo isso muito lentamente, ele será cortado, ou **choked**, mais cedo ou mais tarde. Essa estratégia desencoraja o comportamento antissocial dos free-riders no swarm.

O algoritmo de choking às vezes é descrito como implementando a estratégia **olho por olho**, que encoraja a cooperação em interações repetidas. Porém, ele não impede que os clientes joguem com o sistema em qualquer sentido forte (Piatek et al., 2007). Apesar disso, a atenção ao problema e aos mecanismos que tornam mais difíceis para usuários casuais atuar como free-riders provavelmente contribuiu para o sucesso do BitTorrent.

Como você pode ver pela nossa discussão, o BitTorrent contém um rico vocabulário. Existem torrents, swarms, leechers, seeders e trackers, bem como snubbing, choking, lurking e outros. Para obter mais informações, consulte o pequeno artigo sobre BitTorrent (Cohen, 2003) e procure na Web começando com www.bittorrent.org.

DHTs — Distributed Hash Tables

O surgimento das redes de compartilhamento de arquivos P2P, por volta de 2000, gerou muito interesse na comunidade de pesquisa. A essência dos sistemas P2P é que eles evitam as estruturas gerenciadas de forma central das CDNs e outros sistemas. Essa pode ser uma vantagem significativa. Os componentes gerenciados de forma central se tornam um gargalo quando o sistema fica muito grande, e são um ponto de falha isolado. Os componentes centrais também podem ser usados como um ponto de controle (por exemplo, para encerrar a rede P2P). Contudo, os primeiros sistemas P2P eram apenas parcialmente descentralizados ou, se fossem totalmente descentralizados, seriam ineficazes.

A forma tradicional do BitTorrent que acabamos de descrever utiliza transferências peer-to-peer e um tracker centralizado para cada swarm. É o tracker que se torna a parte difícil de descentralizar em um sistema peer-to-peer. O problema principal é como descobrir quais peers possuem o conteúdo específico que está sendo procurado. Por exemplo, cada usuário poderia ter um ou mais itens de dados, como músicas, fotografias, programas, arquivos etc., que outros usuários poderiam querer ler. Como os outros usuários os encontram? Criar um índice de quem tem o que é simples, mas é centralizado. Fazer com que cada par mantenha seu próprio índice também não ajuda. É verdade que isso é distribuído. Entretanto, exige tanto trabalho para manter atualizados os índices de todos os peers (pois o conteúdo é movimentado pelo sistema) que não compensa o esforço.

A pergunta a ser respondida pela comunidade de pesquisa foi: seria possível criar índices P2P que fossem totalmente distribuídos, mas que tivessem um bom desempenho? Com bom desempenho, queremos dizer três coisas. Primeiro, que cada nó mantém apenas uma pequena quantidade de informação sobre outros nós. Essa propriedade significa que não será caro manter o índice atualizado. Segundo, cada nó pode pesquisar entradas no índice rapidamente. Caso contrário, esse não é um índice muito útil. Terceiro, cada nó pode usar o índice ao mesmo tempo, mesmo que outros nós apareçam e desapareçam. Essa propriedade significa que o desempenho do índice aumenta com o número de nós.

A resposta para a pergunta foi: 'Sim'. Quatro soluções foram inventadas em 2001. Elas são Chord (Stoica et al., 2001), CAN (Ratnasamy et al., 2001), Pastry (Rowstron e Druschel, 2001) e Tapestry (Zhao et al., 2004). As soluções são conhecidas como **DHTs** (**Distributed Hash Tables**), pois a funcionalidade básica de um índice é mapear uma chave a um valor. Isso é como uma tabela de hash, e as soluções são versões distribuídas, naturalmente.

As DHTs realizam seu trabalho impondo uma estrutura regular sobre a comunicação entre os nós, conforme veremos. Esse comportamento é muito diferente daquele das redes P2P tradicionais, que usam quaisquer conexões que os peers façam. Por esse motivo, as DHTs são chamadas **redes P2P estruturadas**. Os protocolos P2P tradicionais montam **redes P2P desestruturadas**.

A solução DHT que descreveremos é a Chord. Como um cenário, considere como substituir o tracker centralizado tradicionalmente usado no BitTorrent por um tracker totalmente distribuído. A Chord pode ser usada para resolver esse problema. Nesse cenário, o índice geral é uma listagem de todos os swarms aos quais um computador pode se juntar para baixar conteúdo. A chave usada para pesquisar o índice é a descrição do conteúdo pela torrent. Ela identifica um swarm exclusivamente, do qual o conteúdo pode ser baixado como os hashes de todos os chunks de conteúdo. O valor armazenado no índice para cada chave é a lista de peers que compreendem o swarm. Esses peers são os computadores a entrar em contato para baixar o conteúdo. Uma pessoa esperando baixar conteúdo como um filme tem apenas a descrição da torrent. A pergunta que a DHT precisa responder é: como, sem um banco de dados central, uma pessoa descobre de quais peers (entre os milhões de nós BitTorrent) ela deve baixar o filme?

Uma Chord DHT consiste em n nós participantes. Eles são nós executando BitTorrent em nosso cenário. Cada nó tem um endereço IP pelo qual pode ser comunicado.

O índice geral está espalhado pelos nós. Isso significa que cada um armazena bits e pedaços do índice para uso por outros. A parte central da Chord é que ela navega pelo índice usando identificadores em um espaço virtual, e não os endereços IP dos nós ou os nomes de conteúdo, como os filmes. Em conceito, os identificadores são simplesmente números de *m* bits que podem ser arrumados em ordem crescente em um anel.

Para transformar um endereço de nó em um identificador, ele é mapeado para um número de *m* bits usando uma função de hash, chamada *hash*. Chord utiliza SHA-1 para o *hash*. Esse é o mesmo hash que mencionamos ao descrever o BitTorrent. Vamos examiná-lo melhor quando discutirmos sobre criptografia, no Capítulo 8. Por enquanto, basta dizer que ele é apenas uma função que captura uma sequência de bytes de tamanho variável como argumento e produz um número de 160 bits altamente aleatório. Assim, podemos usá-lo para converter qualquer endereço IP em um número de 160 bits, chamado **identificador de nó**.

Na Figura 7.41(a), mostramos o anel identificador de nó para $m = 5$. (Basta ignorar os arcos no meio, por enquanto.) Alguns dos identificadores correspondem a nós, mas a maioria não. Neste exemplo, os nós com identificadores 1, 4, 7, 12, 15, 20 e 27 correspondem aos nós reais e estão sombreados na figura; o restante não existe.

Agora, vamos definir a função *sucessor*(*k*) como o identificador de nó do primeiro nó real após *k* ao redor do anel, em sentido horário. Por exemplo, *sucessor* (6) = 7, *sucessor* (8) = 12 e *sucessor* (22) = 27.

Uma **chave** também é produzida pela execução do hashing de um nome de conteúdo com *hash* (ou seja, SHA-1) para gerar um número de 160 bits. Em nosso exemplo, o nome do conteúdo é a torrent. Assim, para converter a *torrent* (o arquivo de descrição de torrent) para sua chave, calculamos *chave* = *hash*(*torrent*). Esse cálculo é simplesmente uma chamada de procedimento local para *hash*.

Para iniciar um novo swarm, um nó precisa inserir um novo par chave-valor consistindo em (*torrent*, *meu-endereço-IP*) no índice. Para conseguir isso, o nó pede a *sucessor*(*hash*(*torrent*)) para armazenar *meu-endereço-IP*. Desse modo, o índice é distribuído pelos nós aleatoriamente. Para conseguir a tolerância a falhas, *p*, diferentes funções de hash poderiam ser usadas para armazenar os dados em *p* nós, mas não vamos considerar o assunto de tolerância a falhas com mais detalhes aqui.

Início	Endereço IP do sucessor
2	4
3	4
5	7
9	12
17	20

Início	Endereço IP do sucessor
5	7
6	7
8	12
12	12
20	20

Início	Endereço IP do sucessor
13	15
14	15
16	20
20	20
28	1

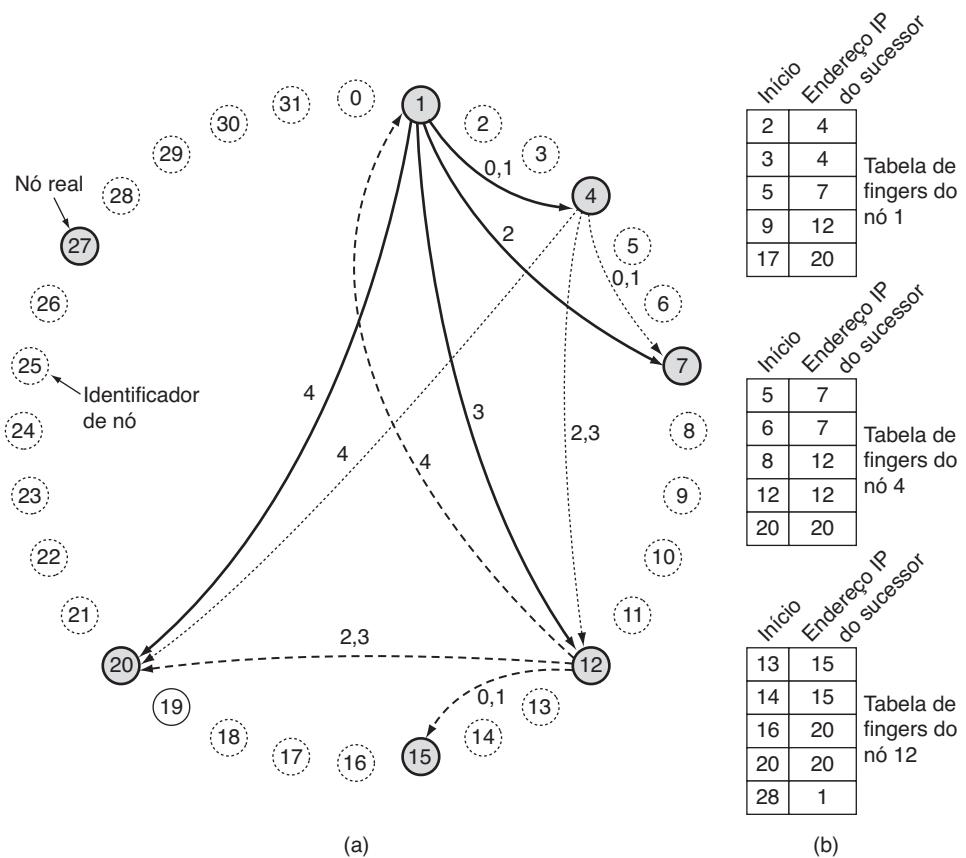



Figura 7.41 | (a) Um conjunto de 32 identificadores de nós arrumados em um anel. Os sombreados correspondem a máquinas reais. Os arcos mostram os fingers dos nós 1, 4 e 12. Os rótulos nos arcos são os índices da tabela. (b) Exemplos de tabelas de fingers.

Algum tempo depois que a DHT é construída, outro nó deseja encontrar uma torrent de modo a se juntar ao swarm e baixar conteúdo. Um nó pesquisa a *torrent* primeiro calculando seu hash para obter a *chave*, e depois usando *sucessor*(*chave*) para encontrar o endereço IP do nó que armazena o valor correspondente. O valor é a lista de peers no swarm; o nó pode acrescentar seu endereço IP na lista e fazer contato com os outros peers para baixar o conteúdo com o protocolo BitTorrent.

A primeira etapa é fácil; a segunda não é tão fácil. Para possibilitar a localização do endereço IP do nó correspondente a uma certa chave, cada nó precisa manter certas estruturas de dados administrativas. Uma delas é o endereço IP do seu nó sucessor junto com o anel identificador de nó. Por exemplo, na Figura 7.41, o sucessor do nó 4 é 7 e o sucessor do nó 7 é 12.

A pesquisa agora pode prosseguir da seguinte forma. O nó solicitante envia um pacote ao seu sucessor, contendo seu endereço IP e a chave que ele está procurando. O pacote é propagado pelo anel até que localize o sucessor do identificador de nó sendo procurado. Esse nó verifica se tem qualquer informação que combine com a chave e, se tiver, a retorna diretamente ao nó solicitante, cujo endereço IP ele possui.

Porém, a busca linear de todos os nós é muito ineficaz em um grande sistema peer-to-peer, pois o número médio de nós exigidos por consulta é $n/2$. Para agilizar bastante a busca, cada nó também mantém o que a Chord chama de **tabela de fingers**. A tabela de fingers contém m entradas, indexadas de 0 até $m - 1$, cada uma apontando para um nó real diferente. Cada uma das entradas tem dois campos: *início* e o endereço IP do *sucessor*(*start)*, como mostramos para três nós no exemplo da Figura 7.41(b). Os valores dos campos para a entrada i em um nó com identificador k são:

$$início = k + 2^i \quad (módulo\ \ 2^m)$$

endereço IP do *sucessor*(*start* [*i*])

Observe que cada nó armazena os endereços IP de um número relativamente pequeno de nós e que a maioria destes é muito próxima em termos de identificador de nó.

Usando a tabela de fingers, a pesquisa da *chave* no nó k prossegue da seguinte forma. Se a *chave* ficar entre k e *sucessor*(k), o nó mantendo informações sobre a *chave* é *sucessor*(k) e a busca termina. Caso contrário, a tabela de fingers é varrida para encontrar a entrada cujo campo *início* é o predecessor mais próximo da *chave*. Uma solicitação é então enviada diretamente ao endereço IP nessa tabela de fingers para pedir que continue a busca. Por estar mais próxima da *chave*, mas ainda abaixo dela, há boa chance de que ela será capaz de retornar a resposta com apenas um pequeno número de consultas adicionais. De fato, como cada pesquisa divide ao meio a distância restante até o destino, pode-se mostrar que o número médio de pesquisas é $\log_2 n$.

Em um primeiro exemplo, considere a pesquisa da *chave* = 3 no nó 1. Como o nó 1 sabe que 3 se encontra, entre ele e seu sucessor, 4, o nó desejado é 4 e a busca termina, retornando o endereço IP de 4.

Como um segundo exemplo, considere a pesquisa da *chave* = 16 no nó 1. Como 16 não se encontra entre 1 e 4, a tabela de fingers é consultada. O predecessor mais próximo de 16 é 9, de modo que a solicitação é encaminhada para o endereço IP da entrada 9, a saber, aquela do nó 12. O nó 12 também não sabe a resposta, de modo que ele procura o nó mais próximo anterior a 16 e encontra 14, que fornece o endereço IP do nó 15. Uma consulta é então enviada para lá. O nó 15 observa que 16 se encontra entre ele e seu sucessor (20), de modo que retorna o endereço IP de 20 para quem chamou, que segue seu caminho de volta até o nó 1.

Visto que os nós entram e saem o tempo todo, Chord precisa de uma forma de lidar com essas operações. Assumimos que, quando o sistema iniciou sua operação, ele era pequeno o suficiente para que os nós pudessem apenas trocar informações diretamente para criar o primeiro anel e tabelas de fingers. Depois disso, um procedimento automatizado é necessário. Quando um novo nó, r, deseja se conectar, ele precisa entrar em contato com um nó existente e pedir que ele pesquise o endereço IP do *sucessor*(r) para ele. Em seguida, o novo nó pede o predecessor do *sucessor*(r). O novo nó então pede a ambos para inserir r entre eles e o anel. Por exemplo, se 24 na Figura 7.41 quiser se conectar, ele pede a qualquer nó para pesquisar o *sucessor*(24), que é 27. Depois ele pede o predecessor de 27, que é 20. Depois de informar a ambos sobre a sua existência, 20 usa 24 como seu sucessor e 27 usa 24 como seu predecessor. Além disso, o nó 27 entrega as chaves no intervalo 21-24, que agora pertencem a 24. Nesse ponto, 24 está totalmente inserido.

Contudo, muitas tabelas de fingers agora estão erradas. Para corrigi-las, cada nó executa um processo em segundo plano que recalcula periodicamente cada finger chamando o *sucessor*. Quando uma dessas consultas alcança um novo nó, a entrada de finger correspondente é atualizada.

Quando um nó sai de modo controlado, ele entrega suas chaves ao seu sucessor e informa ao seu predecessor sobre sua saída, para que ele possa se vincular ao sucessor do nó que está saindo. Quando um nó falha, surge um problema, pois seu predecessor não tem mais um sucessor válido. Para aliviar esse problema, cada nó registra não apenas seu sucessor direto, mas também seus s sucessores diretos, para permitir que ele pule até $s - 1$ nós consecutivos com falhas e reconecte o anel, em caso de desastre.

Tem havido uma grande quantidade de pesquisa sobre DHTs desde que elas foram inventadas. Para que você tenha uma ideia melhor, vamos levantar uma questão: qual é o artigo de rede mais citado em todos os tempos? Você achará difícil chegar a um artigo que seja citado mais do que o original artigo Chord (Stoica et al., 2001). Apesar dessa verdadeira montanha de pesquisa, as aplicações de DHTs estão apenas começando a surgir lentamente. Alguns

clientes BitTorrent utilizam DHTs para fornecer um tracker totalmente distribuído do tipo que descrevemos. Grandes serviços de computação em nuvem, como o Dynamo da Amazon, também incorporam técnicas de DHT (DeCandia et al., 2007).

7.6 Resumo

A atribuição de nomes na ARPANET começou de uma maneira muito simples: um arquivo de texto ASCII listava os nomes de todos os hosts e seus endereços IP correspondentes. A cada noite, todas as máquinas baixavam esse arquivo. Porém, quando a ARPANET se transformou na Internet e explodiu em tamanho, um esquema de nomes bem mais sofisticado e dinâmico foi necessário. Agora, usa-se um esquema hierárquico chamado sistema de nomes de domínios, ou DNS (Domain Name System). Ele organiza todas as máquinas da Internet em um conjunto de árvores. No nível superior, encontram-se os domínios genéricos conhecidos, inclusive *com* e *edu*, bem como cerca de 200 domínios de países. O DNS é implementado sob a forma de um sistema de bancos de dados distribuídos, com servidores espalhados em todo o mundo. Ao consultar um servidor DNS, um processo pode mapear um nome de domínio na Internet no endereço IP usado para comunicação com esse domínio.

O correio eletrônico é a aplicação original mais popular da Internet. Ele é muito usado por todos, desde as crianças até seus avós. A maioria dos sistemas de correio eletrônico no mundo emprega o sistema de correio definido nas RFCs 5321 e 5322. As mensagens possuem cabeçalhos ASCII simples, e muitos tipos de conteúdo podem ser enviados usando o MIME. O e-mail é submetido a agentes de transferência de mensagem para entrega e recuperação, para apresentação por uma série de agentes do usuário, incluindo aplicações Web. As mensagens são enviadas com o uso do SMTP, que funciona estabelecendo uma conexão TCP do agente de transferência de mensagem de saída para o receptor.

A Web é a aplicação que a maioria das pessoas imagina como sendo a Internet. Originalmente, ela era um sistema para vincular páginas de hipertexto (escritas em HTML) de forma transparente entre as máquinas. As páginas são baixadas fazendo uma conexão TCP do navegador para um servidor e usando HTTP. Atualmente, grande parte do conteúdo na Web é produzido dinamicamente, seja no servidor (por exemplo, com PHP), seja no navegador (por exemplo, com JavaScript). Quando combinadas com bancos de dados back-end, páginas dinâmicas do servidor permitem o uso de aplicações Web como e-commerce e busca. Páginas dinâmicas do navegador estão evoluindo para aplicações completas, como correio eletrônico, que são executadas dentro do navegador e usam os protocolos da Web para se comunicar com servidores remotos.

O caching e as conexões persistentes são muito usadas para melhorar o desempenho da Web. O uso da Web em dispositivos móveis pode ser desafiador, apesar do crescimento na largura de banda e do poder de processamento dos dispositivos móveis. Os sites normalmente enviam versões adaptadas de páginas com imagens menores e navegação menos complexa para dispositivos com telas pequenas.

Os protocolos da Web estão cada vez mais sendo usados para a comunicação entre máquinas. XML é preferida à HTML como uma descrição do conteúdo facilitado para as máquinas processarem. SOAP é um mecanismo RPC que envia mensagens XML usando HTTP.

Áudio e vídeo digital têm sido fatores-chave para a Internet desde o ano 2000. A maior parte do tráfego da Internet hoje é vídeo. Grande parte dele flui de sites por uma mistura de protocolos (incluindo RTP/UDP e RTP/HTTP/TCP). A mídia ao vivo flui para muitos consumidores. Ela inclui estações de rádio e TV pela Internet, que enviam todo tipo de evento por broadcast. Áudio e vídeo também são usados para conferências em tempo real. Muitas chamadas usam voz sobre IP, em vez da rede telefônica tradicional, e incluem videoconferência.

Existe um pequeno número de sites muito populares, bem como um grande número de sites pouco populares. Para atender aos mais populares, foram implantadas redes de distribuição de conteúdo. As CDNs usam DNS para direcionar os clientes para um servidor próximo; os servidores ficam espalhados por centros de dados no mundo inteiro. Como alternativa, as redes P2P permitem que uma coleção de máquinas compartilhe conteúdo (como filmes) entre elas. Elas oferecem uma capacidade de distribuição de conteúdo que aumenta com o número de máquinas na rede P2P e que pode competir com o maior dos sites.

Problemas

1. Muitos computadores comerciais têm três identificadores distintos e exclusivos em âmbito mundial. Quais são eles?
2. No Quadro 7.1, não existe nenhum ponto depois de *laserjet*. Por quê?
3. Considere uma situação em que um cyberterrorista faz todos os servidores de DNS do mundo falhar simultaneamente. Como isso muda nossa capacidade de usar a Internet?
4. O DNS utiliza o UDP em vez do TCP. Se um pacote DNS for perdido, não haverá nenhuma recuperação automática. Isso causará problema? Em caso afirmativo, como ele será resolvido?
5. John deseja ter um nome de domínio original e usa um programa de geração aleatória para criar um nome de domínio secundário para ele. John deseja registrar esse nome de domínio no domínio genérico *com*. O nome de domínio que

foi gerado tem 253 caracteres de extensão. O registrador *com* permitirá que esse nome de domínio seja registrado?

6. Uma máquina com um único nome DNS pode ter vários endereços IP? Como isso poderia ocorrer?
7. O número de empresas com um site Web cresceu de modo explosivo nos últimos anos. Como resultado, milhares de empresas se registraram no domínio *com*, provocando uma carga pesada sobre o servidor de nível superior para esse domínio. Sugira um modo de atenuar esse problema sem alterar o esquema de nomenclatura (isto é, sem introduzir novos nomes de domínios de nível superior). Sua solução pode exigir mudanças no código do cliente.
8. Alguns sistemas de correio eletrônico aceitam um campo de cabeçalho *Content Return:*. Esse campo especifica se o corpo da mensagem deve ser retornado caso não seja entregue. Esse campo pertence ao envelope ou ao cabeçalho?
9. Os sistemas de correio eletrônico precisam de diretórios para que os endereços eletrônicos das pessoas possam ser pesquisados. Para criar esses diretórios, os nomes devem ser divididos em componentes-padrão (por exemplo, primeiro nome, último nome) para possibilitar a pesquisa. Descreva alguns problemas que devem ser resolvidos para que um padrão mundial possa ser aceito.
10. Uma grande firma de advocacia, com muitos funcionários, oferece um endereço de e-mail único para cada funcionário. O endereço de e-mail de cada funcionário é login@nomefirma.com. Porém, a firma não definiu explicitamente o formato do login. Assim, alguns funcionários usam seus primeiros nomes como nomes de login, alguns usam seus sobrenomes e outros usam suas iniciais etc. A firma agora deseja criar um formato fixo, por exemplo:

    nome.sobrenome@nomefirma.com,

    que pode ser usado para os endereços de e-mail de todos os seus funcionários. Como isso pode ser feito sem causar muita confusão?
11. Um arquivo binário tem 4.560 bytes. Que tamanho ele terá se for codificado com a técnica de base64, com um par CR+LF inserido após cada 110 bytes enviados e no final?
12. Cite cinco tipos MIME não listados no livro. Você poderá verificar seu navegador ou consultar a Internet para obter informações.
13. Suponha que você queira enviar um arquivo MP3 a um amigo, mas o ISP do seu amigo limita a quantidade de correio recebida a 1 MB e o arquivo MP3 tem 4 MB. Existe algum modo de lidar com essa situação usando a RFC 5322 e o MIME?
14. Suponha que John tenha configurado um mecanismo de autoencaminhamento em seu endereço de e-mail do trabalho, que recebe todos os seus e-mails de trabalho, para encaminhá-los para seu endereço de e-mail pessoal, que ele compartilha com sua esposa. A esposa de John não sabe disso, e ativou um agente de férias em sua conta pessoal. Como John encaminhou seu e-mail, ele não preparou um agente de férias em sua máquina de trabalho. O que acontece quando um e-mail é recebido no endereço de e-mail do trabalho de John?
15. Em qualquer padrão, como o RFC 5322, é necessário haver uma gramática exata do que é permitido, para que diferentes implementações possam interoperar. Até mesmo itens simples têm de estar definidos com cuidado. Os cabeçalhos SMTP permitem espaço em branco entre os símbolos. Forneça *duas* definições alternativas plausíveis de espaço em branco entre símbolos.
16. O agente de férias é parte do agente do usuário ou do agente de transferência de mensagens? É claro que ele é definido com a utilização do agente do usuário, mas o agente do usuário realmente envia as respostas? Explique sua resposta.
17. Em uma versão simples do algoritmo Chord para a pesquisa peer-to-peer, as buscas não usam a tabela de fingers. Em vez disso, elas são lineares em torno do anel, em qualquer direção. Um nó pode prever com precisão em qual direção ele deve pesquisar? Explique sua resposta.
18. O IMAP permite que os usuários busquem e baixem mensagens de correio eletrônico de uma caixa de correio remota. Isso significa que o formato interno das caixas de correio tem de ser padronizado, de forma que qualquer programa IMAP no lado cliente possa ler a caixa de correio em qualquer servidor de correio? Explique sua resposta.
19. Considere o anel Chord da Figura 7.41. Suponha que o nó 18 de repente fique on-line. Qual (is) das tabelas de fingers mostradas na figura é (são) afetada (s)? Como?
20. O Webmail utiliza POP3, IMAP ou nenhum deles? Se ele utiliza algum desses protocolos, por que tal protocolo foi escolhido? Se não usa nenhum deles, qual desses protocolos tem mais afinidades com o Webmail?
21. Quando são enviadas, as páginas Web são prefixadas por cabeçalhos MIME. Por quê?
22. É possível um usuário clicar em um link com o Firefox para abrir um auxiliar específico e clicar no mesmo link no Internet Explorer abrindo um auxiliar completamente diferente, embora o tipo MIME retornado em ambos os casos seja idêntico? Explique sua resposta.
23. Embora não tenha sido mencionada no texto, uma forma alternativa para um URL é o uso do endereço IP em lugar do seu nome DNS. Use essa informação para explicar por que um nome DNS não pode terminar com um dígito.
24. Imagine que alguém no departamento de ciência da computação de Stanford acabou de criar um novo programa e deseja distribuí-lo por FTP para seus colegas analisarem. Essa pessoa coloca o programa no diretório FTP *ftp/pub/paraAnalisar/novaProva.pdf*. Qual deve ser o URL provável para esse programa?
25. Na Tabela 7.10, www.aportal.com mantém as preferências do usuário em um cookie. Uma desvantagem desse esquema é que os cookies se limitam a 4 KB; assim, se as preferências forem extensas — por exemplo, muitas ações, equipes esportivas, tipos de manchetes, previsão do tempo para várias cidades, detalhes especiais para diversas categorias de produtos e outras —, o limite de 4 KB poderá ser alcançado. Crie uma forma alternativa para controlar as preferências que não tenha esse problema.
26. O Banco Preguiça deseja facilitar as transações bancárias on-line para seus clientes preguiçosos, de forma que, após um cliente assinar, ele possa se associar e ser autenticado por uma senha, e o banco retorne um cookie contendo

um número de identificação do cliente. Desse modo, o cliente não terá de se identificar ou digitar uma senha em visitas futuras ao banco on-line. O que você acha dessa ideia? Ela funcionará? É uma boa ideia?

27. (a) Considere a seguinte tag HTML:

<h1> title = <"este é o cabeçalho"> HEADER 1 </h1>

Sob quais condições o navegador usa o atributo *TITLE*, e como?

(b) Qual é a diferença entre o atributo *TITLE* e o atributo *ALT*?

28. Como você torna uma imagem clicável em HTML? Dê um exemplo.

29. Escreva uma página HTML que inclua um link para o endereço de e-mail nomeusuário@NomeDomínio.com. O que acontece quando um usuário clica nesse link?

30. Escreva uma página XML para um registrador de universidade listando vários alunos, cada um com um nome, um endereço e um GPA.

31. Para cada uma das aplicações a seguir, informe se seria (1) possível e (2) melhor usar um script PHP ou JavaScript e por quê.

(a) Exibir um calendário para qualquer mês solicitado desde setembro de 1752.

(b) Exibir a programação de voos de Amsterdã até Nova York.

(c) Criar um grafo para representar um polinômio a partir de coeficientes fornecidos pelo usuário.

32. Escreva um programa em JavaScript que aceite um inteiro maior que 2 e diga se ele é um número primo. Observe que JavaScript tem instruções if e while com a mesma sintaxe de C e Java. O operador de módulo é %. Se precisar da raiz quadrada de *x*, use *Math.sqrt*(*x*).

33. Aqui está uma página HTML:

<html> <body>

 Click here for info

</body> </html>

Se o usuário clicar no hiperlink, será aberta uma conexão TCP e uma série de linhas será enviada ao servidor. Liste todas as linhas enviadas.

34. O cabeçalho *If-Modified-Since* pode ser usado para verificar se uma página guardada em cache ainda é valida. Podem ser feitas solicitações de páginas contendo imagens, som, vídeo e assim por diante, bem como HTML. Você imagina que a eficiência dessa técnica é melhor ou pior para imagens JPEG, em comparação com a HTML? Pense com cuidado no que significa 'eficácia' e explique sua resposta.

35. No dia de um grande evento esportivo, como o jogo do campeonato de algum esporte popular, muitas pessoas visitam o site oficial. Isso é um sucesso instantâneo no mesmo sentido da eleição de 2000 na Flórida? Por quê?

36. Faz sentido um único ISP funcionar como uma CDN? Nesse caso, como ele funcionaria? Se não, o que está errado com essa ideia?

37. Suponha que a compressão não seja usada para CDs de áudio. Quantos MB de dados o CD deve conter a fim de poder tocar duas horas de música?

38. Na Figura 7.16(c), o ruído de quantização ocorre devido ao uso de amostras de 4 bits para representar nove valores de sinais. A primeira amostra no tempo 0 é exata, mas as outras não. Qual é a porcentagem de erros para as amostras obtidas a 1/32, 2/32 e 3/32 do período?

39. Um modelo psicoacústico poderia ser usado para reduzir a largura de banda necessária para telefonia da Internet? Nesse caso, que condições, se houver, teriam de ser atendidas para fazê-lo funcionar? Se não, por que não?

40. Um servidor de streaming de áudio está a uma 'distância' de cerca de 100 ms em relação a um reprodutor de mídia. Ele transmite a saída a 1 Mbps. Se o reprodutor de mídia tiver um buffer de 2 MB, o que você poderá dizer sobre a posição da marca de nível inferior e da marca de nível superior?

41. A voz sobre IP tem os mesmos problemas com firewalls que o streaming de áudio? Explique sua resposta.

42. Qual é a taxa de bits para transmissão de quadros em cores não compactados de 1200 × 800 pixels com 16 bits/pixel a 50 quadros por segundo?

43. Um erro de 1 bit em um quadro MPEG pode afetar outros quadros além daquele no qual o erro ocorre? Explique sua resposta.

44. Considere um servidor de vídeo com 50 mil clientes, em que cada cliente assiste a três filmes por mês. Suponha que dois terços dos filmes sejam transmitidos às 21 horas. Quantos filmes o servidor tem de transmitir ao mesmo tempo durante esse período? Se cada filme exigir 6 Mbps, quantas conexões OC-12 o servidor precisará ter para a rede?

45. Suponha que a lei de Zipf seja válida para acessos a um servidor de vídeo com 10 mil filmes. Supondo que o servidor mantenha os mil filmes mais populares em memória e os outros 9 mil em disco, forneça uma expressão que indique a fração de todas as referências feitas à memória. Crie um pequeno programa para avaliar essa expressão numericamente.

46. Alguns cybersquatters registraram nomes de domínios que são grafias erradas de sites corporativos comuns como, por exemplo, www.microsfot.com. Faça uma lista de pelo menos cinco desses domínios.

47. Várias pessoas registraram nomes DNS que consistem em www.palavra.com, onde palavra é uma palavra comum. Para cada uma das categorias a seguir, liste cinco sites desse tipo e faça um breve resumo de sua finalidade (por exemplo, www.estomacal.com é um gastroenterologista de Niterói). Aqui está a lista de categorias: animais, comidas, objetos domésticos e partes do corpo. Para a última categoria, por favor, tome como exemplos partes do corpo acima da cintura.

48. Reescreva o servidor da Figura 6.6 como um verdadeiro servidor Web usando o comando *GET* do HTTP 1.1. Ele também deve aceitar a mensagem *Host*. O servidor deve manter um cache de arquivos recentemente buscados no disco e atender a solicitações de caching, quando possível.

Segurança de redes

Capítulo 8

Durante as primeiras décadas de sua existência, as redes de computadores foram usadas principalmente por pesquisadores universitários, para enviar mensagens de correio eletrônico, e também por funcionários de empresas, para compartilhar impressoras. Nessas condições, a segurança nunca precisou de maiores cuidados. Porém, como milhões de cidadãos comuns atualmente usam as redes para executar operações bancárias, fazer compras e arquivar sua devolução de impostos, surge um ponto fraco atrás de outro, e a segurança tornou-se um problema de grandes proporções. Neste capítulo, estudaremos a segurança das redes sob vários ângulos, destacaremos muitas falhas e discutiremos diversos algoritmos e protocolos que as tornam mais seguras.

A segurança é um assunto abrangente e inclui inúmeros tipos de problemas. Em sua forma mais simples, preocupa-se em impedir que pessoas mal-intencionadas leiam ou, pior ainda, modifiquem secretamente mensagens enviadas a outros destinatários. Outra preocupação da segurança são as pessoas que tentam ter acesso a serviços remotos que não estão autorizadas a usar. Ela também lida com meios para saber se uma mensagem supostamente verdadeira é um trote. A segurança trata de situações em que mensagens legítimas são capturadas e reproduzidas, além de lidar com pessoas que tentam negar o fato de ter enviado certas mensagens.

A maior parte dos problemas de segurança é causada propositalmente por pessoas mal-intencionadas, que tentam obter algum benefício, chamar atenção ou prejudicar alguém. Alguns dos invasores mais comuns estão listados na Tabela 8.1. Fica claro por essa lista que tornar uma rede segura envolve muito mais que apenas mantê-la livre de erros de programação. Para tornar uma rede segura, com frequência é necessário lidar com adversários inteligentes, dedicados e, às vezes, muito bem subsidiados. Você também deverá ter em mente que as medidas utilizadas para interromper a atividade de adversários eventuais terão pouco impacto sobre os adversários 'mais espertos'. Os registros policiais mostram que a maioria dos ataques não é perpetrada por estranhos que grampeiam uma linha telefônica, mas por pessoas ressentidas com a organização a que pertencem. Os sistemas de segurança devem ser projetados tendo em vista esse fato.

Adversário	Objetivo
Estudante	Divertir-se bisbilhotando as mensagens de correio eletrônico de outras pessoas
Cracker	Testar o sistema de segurança de alguém; roubar dados
Representante de vendas	Tentar representar toda a Europa e não apenas Andorra
Executivo	Descobrir a estratégia de marketing do concorrente
Ex-funcionário	Vingar-se por ter sido demitido
Contador	Desviar dinheiro de uma empresa
Corretor de valores	Negar uma promessa feita a um cliente por meio de uma mensagem de correio eletrônico
Vigarista	Roubar números de cartão de crédito e vendê-los
Espião	Descobrir segredos militares ou industriais de um inimigo
Terrorista	Roubar segredos de armas bacteriológicas

Tabela 8.1 | Algumas pessoas que podem causar problemas de segurança e os motivos para fazê-lo.

Os problemas de segurança das redes podem ser divididos nas seguintes áreas interligadas: sigilo, autenticação, não repúdio e controle de integridade. O sigilo — também chamado confidencialidade — está relacionado ao fato de manter as informações longe de usuários não autorizados. É isso que costuma nos vir à mente quando pensamos em segurança de redes. Em geral, a autenticação cuida do processo de determinar com quem você está se comunicando antes de revelar informações sigilosas ou entrar em uma transação comercial. O não repúdio trata de assinaturas: como provar que seu cliente realmente fez um pedido eletrônico de dez milhões de unidades de um produto com preço unitário de 89 centavos quando mais tarde ele afirmar que o preço era 69 centavos? Ou talvez ele afirme que nunca efetuou nenhum pedido. Por fim, como você pode se certificar de que uma mensagem recebida é de fato legítima e não algo que um oponente mal-intencionado modificou a caminho ou inventou?

Todas essas questões (sigilo, autenticação, não repúdio e controle de integridade) também ocorrem em sistemas

tradicionais, mas com algumas diferenças significativas. O sigilo e a integridade são obtidos graças à utilização de correspondência registrada e de bloqueio de documentos. Hoje é mais difícil roubar o trem postal do que nos tempos de Jesse James.

Além disso, é comum as pessoas conseguirem distinguir um documento original de uma fotocópia, e isso quase sempre faz diferença para elas. Como teste, tire uma fotocópia de um cheque válido. Tente descontar o cheque original na segunda-feira. Agora tente descontar a fotocópia do cheque na terça-feira. Observe a diferença no comportamento do caixa. Já com os cheques eletrônicos, não há como distinguir entre o original e a cópia. Talvez leve algum tempo até que os bancos aprendam a lidar com isso.

As pessoas autenticam outras pessoas por vários meios, incluindo reconhecimento de rosto, voz e caligrafia. As comprovações de assinatura são feitas por meio de papel timbrado, de símbolos em alto-relevo etc. Em geral, as falsificações podem ser detectadas por especialistas em caligrafia, papel e tinta. Nenhuma dessas opções está disponível eletronicamente. É claro que são necessárias outras soluções.

Antes de entrarmos nas soluções propriamente ditas, vale a pena dedicar alguns momentos considerando a que parte da pilha de protocolos pertence a segurança de redes. É provável que não exista uma parte específica. Todas as camadas contribuem de alguma forma. Na camada física, os 'grampos' podem ser anulados mantendo-se as linhas de transmissão (ou seja, as fibras ópticas) em tubos lacrados contendo gás inerte em alta pressão. Qualquer tentativa de perfurar um tubo liberará o gás, reduzindo a pressão e disparando um alarme. Alguns sistemas militares utilizam essa técnica.

Na camada de enlace de dados, os pacotes de uma linha ponto a ponto podem ser codificados à medida que saem de uma máquina, e decodificados quando entram em outro sistema. Todos os detalhes podem ser tratados na camada de enlace de dados, com as camadas mais altas alheias ao que está acontecendo. No entanto, essa solução se mostra ineficiente quando os pacotes têm de atravessar vários roteadores, pois é necessário descriptografar os pacotes em cada roteador, o que os torna vulneráveis a ataques dentro do roteador. Além disso, essa estratégia permite que algumas sessões sejam protegidas (por exemplo, aquelas que envolvem compras on-line por cartão de crédito) e outras não. Todavia, a **criptografia no nível do enlace de dados**, como esse método é chamado, pode ser facilmente incluída em qualquer rede e com frequência é muito útil.

Na camada de rede, podem ser instalados firewalls para manter ou descartar pacotes. A segurança do IP também funciona nessa camada.

Na camada de transporte, é possível criptografar conexões inteiras ponto a ponto, ou seja, processo a processo. Para obter segurança máxima, é necessário que ela seja ponto a ponto.

Por fim, questões como autenticação do usuário e não repúdio só podem ser tratadas na camada de aplicação.

Tendo em vista que a segurança não se ajusta nitidamente a nenhuma camada, ela também não se enquadra em nenhum capítulo deste livro. Por essa razão, merece seu próprio capítulo.

Embora este capítulo seja longo, técnico e essencial, isso é quase irrelevante no momento. Por exemplo, está bem documentado o fato de que a maioria das falhas de segurança em bancos se deve a funcionários incompetentes, procedimentos de segurança negligentes, diversos bugs de implementação que permitem a invasão remota por usuários não autorizados e os chamados ataques de engenharia social, nos quais os clientes são enganados para revelar os detalhes de sua conta. Todos esses problemas de segurança acontecem com mais frequência do que a ação de criminosos inteligentes grampeando linhas telefônicas e depois decodificando mensagens criptografadas. Se uma pessoa puder entrar em uma agência bancária qualquer com uma tira de extrato de caixa eletrônico que encontrou na rua, afirmando que esqueceu sua senha e receber uma nova senha na mesma hora (em nome das boas relações com os clientes), nem toda criptografia do mundo evitará o abuso. Sobre esse aspecto, o livro de Ross Anderson (2008a) é um excelente alerta, pois documenta centenas de exemplos de falhas de segurança em numerosos setores, quase todas causadas por aquilo que se poderia chamar educadamente de práticas comerciais relaxadas ou desatenção com a segurança. Apesar disso, a base técnica sobre a qual o comércio eletrônico é montado, quando todos esses outros fatores são bem elaborados, é a criptografia.

Com exceção da segurança na camada física, quase toda segurança se baseia em princípios criptográficos. Por essa razão, começaremos nosso estudo da segurança examinando em detalhes a criptografia. Na Seção 8.1, veremos alguns princípios básicos. Da Seção 8.2 até a Seção 8.5, examinaremos alguns algoritmos e estruturas de dados fundamentais usados na criptografia. Em seguida, examinaremos em detalhes como esses conceitos podem ser usados para alcançar a segurança em redes. Concluiremos com alguns conceitos breves sobre tecnologia e sociedade.

Antes de começarmos, devemos chamar atenção para o que não será abordado neste capítulo. Procuramos nos concentrar em questões de redes, e não naquelas relacionadas ao sistema operacional e às aplicações, embora seja difícil traçar uma linha clara separando esses assuntos. Por exemplo, não há nada aqui sobre autenticação do usuário com a utilização da biometria, segurança de senhas, ataques por overflow de buffers, cavalos de Troia ou trojans, spoofing de login, inserção de código em scripts de sites, vírus, worms e temas semelhantes. Todos esses tópicos são abordados detalhadamente no Capítulo 9 do livro *Modern Operating Systems* (Tanenbaum, 2007). O leitor interessado

deve consultar esse livro para conhecer os aspectos de segurança relacionados aos sistemas. Agora, vamos iniciar nossa jornada.

8.1 Criptografia

A palavra **criptografia** vem de palavras gregas que significam 'escrita secreta'. A criptografia tem uma longa e interessante história de milhares de anos. Nesta seção, vamos esquematizar alguns destaques, que serão usados como informações básicas para o que vem a seguir. Se desejar um histórico completo da criptografia, recomendamos a leitura do livro de Khan (1995). Para ver um tratamento completo do estado da arte em segurança e algoritmos criptográficos, protocolos, aplicações e material relacionado, consulte Kaufman et al. (2002). Para uma abordagem mais matemática, consulte Stinson (2002). Se preferir uma abordagem menos matemática, consulte Burnett e Paine (2001).

Os profissionais fazem distinção entre cifras e códigos. Uma **cifra** é uma transformação de caractere por caractere ou de bit por bit, sem levar em conta a estrutura linguística da mensagem. Em contrapartida, um **código** substitui uma palavra por outra palavra ou símbolo. Os códigos não são mais utilizados, embora tenham uma história gloriosa. O código mais bem-sucedido já inventado foi usado pelas forças armadas dos Estados Unidos durante a Segunda Guerra Mundial no Pacífico. Elas simplesmente tinham índios navajos que se comunicavam uns com os outros usando palavras em Navajo específicas para termos militares, como *chay-dagahi-nail-tsaidi* (literalmente, matador de cágado) para indicar uma arma antitanque. A linguagem navajo é altamente tonal, extremamente complexa, e não tem nenhuma forma escrita. Além disso, nem uma única pessoa no Japão conhecia alguma coisa sobre ela.

Em setembro de 1945, o *San Diego Union* descreveu o código da seguinte forma: "Por três anos, onde quer que os *marines* aterrissassem, os japoneses recebiam uma enxurrada de estranhos ruídos gorgolejantes entremeados com outros sons que lembravam o clamor de um monge tibetano e o som de uma bolsa de água quente sendo esvaziada". Os japoneses nunca conseguiram quebrar o código e muitos índios navajos receberam altas honras militares por serviço e bravura extraordinários. O fato de os Estados Unidos terem conseguido quebrar o código japonês e os japoneses nunca terem conseguido quebrar o código navajo desempenhou um papel crucial nas vitórias norte-americanas no Pacífico.

8.1.1 Introdução à criptografia

Historicamente, quatro grupos de pessoas utilizaram e contribuíram para a arte da criptografia: os militares, os diplomatas, as pessoas que gostam de guardar memórias e os amantes. Entre eles, os militares tiveram o papel mais importante e definiram as bases para a tecnologia. Nas organizações militares, as mensagens a ser criptografadas eram entregues habitualmente a auxiliares mal remunerados, que se encarregavam de criptografá-las e transmiti-las. O grande volume de mensagens impedia que esse trabalho fosse feito por alguns poucos especialistas.

Até o advento dos computadores, uma das principais restrições impostas à criptografia era a habilidade do auxiliar de criptografia para fazer as transformações necessárias, em geral com poucos equipamentos e no campo de batalha. Outra restrição era a dificuldade de alternar os métodos criptográficos rapidamente, pois isso exigia a repetição do treinamento de um grande número de pessoas. No entanto, o perigo de um auxiliar de criptografia ser capturado pelo inimigo tornou indispensável a necessidade de alterar o método criptográfico instantaneamente, se preciso. Essas necessidades conflitantes fizeram surgir o modelo da Figura 8.1.

As mensagens a ser criptografadas, conhecidas como **texto simples** (ou plaintext), são transformadas por meio de uma função parametrizada por uma **chave**. Em seguida,

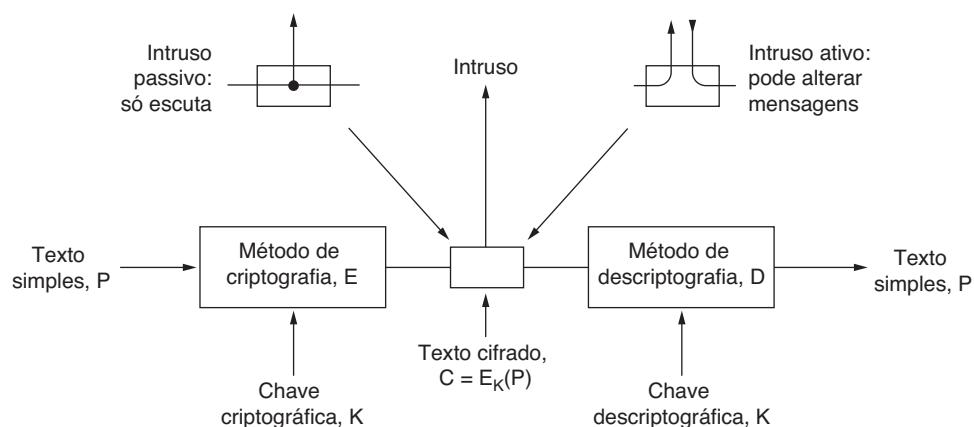



Figura 8.1 | O modelo de criptografia (para uma cifra de chave simétrica).

a saída do processo de criptografia, conhecida como **texto cifrado** (ou ciphertext), é transmitida, quase sempre por um mensageiro ou por rádio. Presumimos que o inimigo, ou **intruso**, ouça e copie cuidadosamente o texto cifrado completo. No entanto, ao contrário do destinatário pretendido, ele não conhece a chave para descriptografar o texto e, portanto, não pode fazê-lo com muita facilidade. Às vezes, o intruso pode não só escutar o que se passa no canal de comunicação (intruso passivo) como também gravar mensagens e reproduzi-las mais tarde, injetar suas próprias ou modificar mensagens legítimas antes que elas cheguem ao receptor (intruso ativo). A arte de solucionar mensagens cifradas, conhecida como **criptoanálise**, e a arte de criar mensagens cifradas (criptografia) é chamada coletivamente **criptologia**.

Com frequência, será útil e prático ter uma notação para estabelecer uma relação entre o texto simples, o texto cifrado e as chaves. Usaremos $C = E_K(P)$ para indicar que a criptografia do texto simples P usando a chave K gera o texto cifrado C. Da mesma forma, $P = D_K(C)$ representa a descriptografia de C para obter o texto simples outra vez. Então, temos:

$$D_K(E_K(P)) = P$$

Essa notação sugere que E e D são simplesmente funções matemáticas, o que é verdade. A única parte complicada é que ambas são funções de dois parâmetros, e escrevemos um desses parâmetros (a chave) como um caractere subscrito, em vez de representá-lo como um argumento, para distingui-lo da mensagem.

Uma regra fundamental da criptografia é que se deve supor que o criptoanalista conhece os métodos genéricos de criptografia e descriptografia que são utilizados. Em outras palavras, o criptoanalista conhece os detalhes de como funcionam o método de criptografia, E, e o método de descriptografia D da Figura 8.1. O esforço necessário para criar, testar e instalar um novo algoritmo toda vez que o antigo método é (supostamente) comprometido sempre tornou impraticável manter o algoritmo criptográfico em segredo. Imaginar que esse algoritmo é secreto, quando não é, resulta mais em prejuízo do que em benefícios.

É nesse ponto que entra a chave. A chave consiste em uma string (relativamente) curta que seleciona uma das muitas formas possíveis de criptografia. Ao contrário do método genérico, que só pode ser modificado a cada período de alguns anos, a chave pode ser alterada sempre que necessário. Portanto, nosso modelo básico é um método genérico publicamente conhecido, parametrizado por uma chave secreta que pode ser alterada com facilidade. A ideia de que o criptoanalista conhece os algoritmos e que o segredo reside exclusivamente nas chaves é chamada **princípio de Kerckhoff**, em homenagem ao criptógrafo militar flamengo Auguste Kerckhoff que primeiro o enunciou em 1883 (Kerckhoff, 1883). Desse modo, temos:

Princípio de Kerckhoff: Todos os algoritmos devem ser públicos; apenas as chaves são secretas

Devemos enfatizar o caráter não sigiloso do algoritmo. Tentar manter o algoritmo secreto, uma estratégia conhecida no ramo como **segurança pela obscuridade**, nunca funciona. Além disso, ao tornar o algoritmo público, o especialista em criptografia libera a consulta para inúmeros criptólogos ansiosos por decodificar o sistema e assim publicar artigos demonstrando suas habilidades. Caso muitos especialistas tenham tentado decodificá-lo durante cinco anos após sua publicação e nenhum tenha conseguido, isso significa que o algoritmo é provavelmente sólido.

Na verdade, o sigilo está na chave, e seu tamanho é uma questão muito importante do projeto. Considere uma fechadura de combinação simples. Segundo o princípio geral, você insere dígitos em sequência. Todo mundo sabe disso, mas a chave é secreta. Uma chave com um tamanho de dois dígitos significa que existem cem possibilidades. Uma de três dígitos significa mil possibilidades e uma de seis dígitos significa um milhão de possibilidades. Quanto maior for a chave, mais alto será o **fator de trabalho** com que o criptoanalista terá de lidar. O fator de trabalho para decodificar o sistema por meio de uma exaustiva pesquisa no espaço da chave é exponencial em relação a seu tamanho. O sigilo é decorrente da presença de um algoritmo forte (mas público) e de uma chave longa. Para impedir que seu irmãozinho leia suas mensagens de correio eletrônico, serão necessárias chaves de 64 bits. Para uso comercial de rotina, devem ser usados pelo menos 128 bits. Para manter o governo de outros países à distância, são necessárias chaves de pelo menos 256 bits, de preferência maiores.

Do ponto de vista do criptoanalista, o problema da criptoanálise apresenta três variações principais. Quando tem determinado volume de texto cifrado mas nenhum texto simples, o analista é confrontado com o problema do texto cifrado disponível. Os criptogramas da seção de palavras cruzadas do jornal são um exemplo desse tipo de problema. Quando há uma correspondência entre o texto cifrado e o texto simples, o problema passa a ser chamado de **texto simples conhecido**. Por fim, quando o criptoanalista tem a possibilidade de codificar trechos do texto simples escolhidos por ele mesmo, temos o problema do **texto simples escolhido**. Os criptogramas dos jornais poderiam ser decodificados de forma trivial se o criptoanalista tivesse a permissão de fazer perguntas tais como: qual é a criptografia de ABCDEFGHIJKL?

Com frequência, os novatos na área de criptografia pressupõem que, se uma cifra puder resistir a um ataque do texto cifrado disponível, isso significa que ela é segura. Essa suposição é muito ingênua. Em muitos casos, o criptoanalista pode fazer uma estimativa com base em trechos do texto simples. Por exemplo, a primeira mensagem que muitos sistemas de tempo compartilhado emitem quando você os chama é 'login:'. Equipado com alguns pares de

texto simples/texto cifrado, o trabalho do criptoanalista se torna muito mais fácil. Para ter segurança, o autor da criptografia deve ser conservador e se certificar de que o sistema seja inviolável, mesmo que seu oponente seja capaz de criptografar o texto simples escolhido.

Historicamente, os métodos de criptografia têm sido divididos em duas categorias: as cifras de substituição e as cifras de transposição. Em seguida, trataremos de cada uma dessas técnicas como informações básicas para a criptografia moderna.

8.1.2 Cifras de substituição

Em uma **cifra de substituição**, cada letra ou grupo de letras é substituído por outra letra ou grupo de letras, de modo a criar um 'disfarce'. Uma das cifras mais antigas é a **cifra de César**, atribuída a Julio César. Nesse método, *a* se torna *D*, *b* se torna *E*, *c* se torna *F*,... e *z* se torna *C*. Por exemplo, *ataque* passaria a ser *DWDTXH*. Em nossos exemplos, o texto simples é apresentado em minúsculas e o texto cifrado em maiúsculas.

Uma ligeira generalização da cifra de César permite que o alfabeto do texto cifrado seja deslocado *k* letras, em vez de três. Nesse caso, *k* passa a ser uma chave para o método genérico dos alfabetos deslocados em forma circular. A cifra de César pode ter enganado Pompeu, mas nunca mais enganou ninguém.

O próximo aprimoramento é fazer com que cada um dos símbolos do texto simples, digamos, as 26 letras, seja mapeado para alguma outra letra. Por exemplo

texto simples: a b c d e f g h i j k l m n o p q r s t u v w x y z

texto cifrado: Q W E R T Y U I O P A S D F G H J K L Z X C V B N M

Esse sistema geral é chamado **cifra de substituição monoalfabética,** sendo a chave a string de 26 letras correspondente ao alfabeto completo. Para a chave anterior, o texto simples *ataque* seria transformado no texto cifrado *QZQJXT*.

À primeira vista, talvez esse sistema pareça seguro, pois, apesar de conhecer o sistema genérico (substituição de letra por letra), o criptoanalista não sabe qual das $26! \approx 4 \times 10^{26}$ chaves possíveis está em uso. Ao contrário do que acontece com a cifra de César, experimentar todas elas não é uma estratégia muito promissora. Mesmo a 1 ns por solução, um milhão de chips de computador em paralelo demandariam 10 mil anos para que todas as chaves fossem experimentadas.

Todavia, com um volume de texto cifrado surpreendentemente pequeno, a cifra pode ser descoberta com facilidade. A estratégia básica se beneficia das propriedades estatísticas dos idiomas. Por exemplo, em inglês *e* é a letra mais comum, seguida de *t*, *o*, *a*, *n*, *i* etc. As combinações de duas letras, ou **digramas**, mais comuns são *th*, *in*, *er*, *re* e *an*. As combinações de três letras, ou **trigramas**, mais comuns são *the*, *ing*, *and* e *ion*.

Um criptoanalista que tente decodificar uma cifra monoalfabética começaria contando as frequências relativas de todas as letras do texto cifrado. Depois disso, por tentativas, ele atribuiria *e* à letra mais comum e *t* à próxima letra mais comum. Em seguida, verificaria os trigramas para encontrar um no formato *tXe*, o que poderia sugerir que *X* é *h*. Da mesma forma, se o padrão *thYt* ocorrer com frequência, provavelmente isso significará que *Y* representa *a*. Com essas informações, o criptoanalista poderá procurar por um trigrama com o formato *aZW* que ocorra com frequência (muito provavelmente *and*). Fazendo estimativas em relação a digramas, trigramas e letras mais comuns, e conhecendo os prováveis padrões de vogais e consoantes, o criptoanalista criaria um texto simples através de tentativas, letra por letra.

Outra estratégia é adivinhar uma palavra ou frase provável. Por exemplo, considere o seguinte texto cifrado de uma empresa de contabilidade (montado em grupos de cinco caracteres):

```
CTBMN BYCTC BTJDS QXBNS GSTJC BTSWX CTQTZ CQVUJ
QJSGS TJQZZ MNQJS VLNSX VSZJU JDSTS JQUUS JUBXJ
DSKSU JSNTK BGAQJ ZBGYQ TLCTZ BNYBN QJSW
```

Nos Estados Unidos, uma palavra muito provável em uma mensagem de uma empresa de contabilidade é *financial*. Com base em nosso conhecimento de que *financial* tem um caractere repetido (*i*), com quatro outras letras entre suas ocorrências, procuramos letras repetidas no texto cifrado com esse espaço entre elas. Encontramos 12 casos como esse nas posições 6, 15, 27, 31, 42, 48, 56, 66, 70, 71, 76 e 82. No entanto, apenas dois deles, 31 e 42, têm a letra seguinte (que corresponde a *n* no texto simples) repetida na localização adequada. Dessas duas, apenas 31 também tem a letra *a* corretamente posicionada; portanto, sabemos que *financial* começa na posição 30. Desse ponto em diante, fica fácil deduzir a chave utilizando a estatística de frequência para o texto em inglês e procurando palavras quase completas para terminar.

8.1.3 Cifras de transposição

As cifras de substituição preservam a ordem dos símbolos no texto simples, mas os disfarçam. Por outro lado, as **cifras de transposição** reordenam as letras, mas não as disfarçam. A Figura 8.2 mostra uma cifra de transposição muito comum, a de colunas. A cifra se baseia em uma chave que é uma palavra ou frase que não contém letras repetidas. Nesse exemplo, a chave é MEGABUCK. O objetivo da chave é numerar as colunas de modo que a coluna 1 fique abaixo da letra da chave mais próxima do início do

```
M E G A B U C K
7 4 5 1 2 8 3 6
p l e a s e t r
a n s f e r o n
e m i l l i o n
d o l l a r s t
o m y s w i s s
b a n k a c c o
u n t s i x t w
o t w o a b c d
```

Texto simples

pleasetransferonemilliondollarsto
myswissbankaccountsixtwotwo

Texto cifrado

AFLLSKSOSELAWAIATOOSSCTCLNMOMANT
ESILYNTWRNNTSOWDPAEDOBUOERIRICXB

Figura 8.2 | Uma cifra de transposição.

alfabeto e assim por diante. O texto simples é escrito horizontalmente, em linhas. O texto cifrado é lido em colunas, a partir da coluna cuja letra da chave seja a mais baixa.

Para quebrar uma cifra de transposição, o criptoanalista deve primeiro estar ciente de lidar com tal cifra. Ao examinar a frequência de *E*, *T*, *A*, *O*, *I*, *N* etc., fica fácil constatar se essas letras se encaixam no padrão normal para texto simples. Se houver correspondência, isso significa que se trata sem dúvida de uma cifra de transposição, pois nesse tipo de cifra cada letra é representada por ela mesma, mantendo intacta a distribuição de frequências.

A próxima etapa é fazer uma estimativa do número de colunas. Em muitos casos, uma palavra ou frase provável pode ser deduzida a partir do contexto da mensagem. Por exemplo, suponha que nosso criptoanalista tenha suspeitado de que a frase em texto simples *milliondollars* ocorre em algum lugar na mensagem. Observe que os digramas *MO*, *IL*, *LL*, *LA*, *IR* e *OS* ocorrem no texto cifrado como um resultado do desdobramento dessa frase. No texto cifrado, a letra *O* vem depois da letra *M* (ou seja, elas são verticalmente adjacentes na coluna 4), pois estão separadas na provável frase por uma distância igual ao tamanho da chave. Se tivesse sido usada uma chave de tamanho sete, teriam surgido os digramas *MD*, *IO*, *LL*, *LL*, *IA*, *OR* e *NS*. Na verdade, para cada tamanho de chave, é produzido um conjunto de digramas diferente no texto cifrado. Ao tentar encontrar diferentes possibilidades, muitas vezes o criptoanalista é capaz de determinar com facilidade o tamanho da chave.

A última etapa é ordenar as colunas. Quando o número de colunas k é pequeno, cada um dos $k(k - 1)$ pares de colunas pode ser examinado para que se constate se suas frequências de digramas correspondem às do texto simples em inglês. O par que tiver a melhor correspondência será considerado na posição correta. Em seguida, cada uma das colunas restantes é experimentada como sucessora desse par. A coluna cujas frequências de digramas e trigramas proporcionem a melhor correspondência será a título experimental considerada correta. A próxima coluna é encontrada da mesma maneira. O processo inteiro continua até ser encontrada uma ordenação potencial. O mais provável é que o texto simples seja reconhecido nesse ponto (por exemplo, se ocorrer *milloin*, ficará claro qual é o erro).

Algumas cifras de transposição aceitam um bloco de tamanho fixo como entrada e produzem um bloco de tamanho fixo como saída. Essas cifras podem ser completamente descritas fornecendo-se uma lista que informe a ordem na qual os caracteres devem sair. Por exemplo, a cifra da Figura 8.2 pode ser vista como uma cifra de blocos de 64 caracteres. Sua saída é 4, 12, 20, 28, 36, 44, 52, 60, 5, 13,..., 62. Em outras palavras, o quarto caractere de entrada, *a*, é o primeiro a sair, seguido pelo décimo segundo, *f*, e assim por diante.

8.1.4 Chave única

Na verdade, é fácil criar uma cifra inviolável; a técnica é conhecida há décadas. Primeiro, escolha como chave uma sequência de bits aleatórios. Em seguida, converta o texto simples em uma sequência de bits, utilizando por exemplo sua representação ASCII. Por fim, calcule o OU exclusivo (XOR) dessas duas sequências. O texto cifrado resultante não pode ser violado porque, em uma amostra grande o suficiente de texto cifrado, cada letra ocorrerá com a mesma frequência, bem como o digrama, o trigrama e assim por diante. Esse método, conhecido como **chave única**, é imune a todos os ataques presentes e futuros, não importa quanta capacidade computacional tenha o intruso. A razão deriva da teoria da informação: simplesmente não existe nenhuma informação na mensagem, todos os textos simples possíveis com o tamanho dado são igualmente prováveis.

Um exemplo de como as chaves únicas são usadas é dado na Figura 8.3. Primeiro, a mensagem 1, 'I love you', é convertida em ASCII de 7 bits. Em seguida, uma chave única, chamada chave 1, é escolhida e sujeita à operação XOR com a mensagem para obter o texto cifrado. Um criptoanalista poderia experimentar todas as chaves únicas possíveis para ver que texto resultou para cada uma. Por exemplo, a chave única listada como chave 2 na figura poderia ser experimentada, resultando no texto simples 2, 'Elvis lives' [Elvis não morreu], que pode ser ou não plausível (um assunto que está além do escopo deste livro). De fato, para cada texto simples ASCII de 11 caracteres, existe uma chave única que o gera. É isso que queremos dizer quando mencionamos que não existe nenhuma informação no texto cifrado: é possível obter qualquer mensagem com o tamanho correto a partir dele.

As chaves únicas são ótimas na teoria, mas têm várias desvantagens na prática. Para começar, não pode ser memorizada; então, tanto o remetente quanto o destinatário devem levar uma cópia escrita com eles. Se qualquer um dos dois estiver sujeito à possibilidade de captura, as chaves escritas sem dúvida serão inconvenientes. Além disso, a quantidade total de dados que podem ser transmitidos é limitada pelo tamanho da chave disponível. Se o espião tiver muita sorte e descobrir uma grande quantidade de dados, talvez

Mensagem 1:	1001001	0100000	1101100	1101111	1110110	1100101	0100000	1111001	1101111	1110101	0101110
Chave 1:	1010010	1001011	1110010	1010101	1010010	1100011	0001011	0101010	1010111	1100110	0101011
Texto cifrado:	0011011	1101011	0011110	0111010	0100100	0000110	0101011	1010011	0111000	0010011	0000101
Chave 2:	1011110	0000111	1101000	1010011	1010111	0100110	1000111	0111010	1001110	1110110	1110110
Texto simples 2:	1000101	1101100	1110110	1101001	1110011	0100000	1101100	1101001	1110110	1100101	1110011

Figura 8.3 | O uso de uma chave única para criptografia e a possibilidade de conseguir qualquer texto simples que seja possível a partir do texto cifrado pela utilização de alguma outra chave.

seja incapaz de transmiti-los de volta para a matriz, porque a chave terá sido consumida. Outro problema é a sensibilidade do método para caracteres perdidos ou inseridos. Se o transmissor e o receptor ficarem fora de sincronismo, todos os dados a partir desse momento se parecerão com lixo.

Com o advento dos computadores, a chave única se tornou potencialmente prática para algumas aplicações. A origem da chave poderia ser um DVD especial contendo vários gigabytes de informações que, se fossem transportadas em uma caixa de filmes em DVD e tivessem no início alguns minutos de vídeo, nem sequer seriam suspeitas. É claro que, nas redes de gigabits, ter de inserir um novo DVD a cada 30 segundos seria algo entediante. Além disso, os DVDs devem ser transportados em mãos, do transmissor para o receptor, antes de ser possível enviar qualquer mensagem, o que reduz bastante sua utilidade prática.

Criptografia quântica

É interessante, mas talvez haja uma solução para o problema de como transmitir a chave única pela rede, e ela vem de uma fonte muito improvável: a mecânica quântica. Essa área ainda é experimental, mas os testes iniciais são promissores. Se eles puderem ser aperfeiçoados e se tornarem eficientes, quase toda a criptografia será realizada por fim com a utilização de chaves únicas, pois é provável que elas sejam seguras. A seguir, explicaremos em linhas gerais como funciona esse método, denominado **criptografia quântica**. Em particular, descreveremos um protocolo chamado **BB84** para indicar seus autores e o ano da publicação (Bennet e Brassard, 1984).

Uma usuária chamada Alice quer estabelecer uma chave única com um segundo usuário, Bob. Alice e Bob são chamados **protagonistas**, os personagens principais de nossa história. Por exemplo, Bob é um gerente de banco com quem Alice gostaria de realizar negócios. Os nomes 'Alice' e 'Bob' foram usados como protagonistas em praticamente todos os ensaios e livros sobre criptografia desde que Ron Rivest os apresentou há muitos anos (Rivest et al., 1978). Os criptógrafos amam a tradição. Se fôssemos usar 'Andy' e 'Barbara' como protagonistas, ninguém acreditaria em nada do que fosse explicado neste capítulo. Então, que seja!

Se Alice e Bob pudessem estabelecer uma chave única, eles teriam a possibilidade de empregá-la para se comunicar com segurança. A pergunta é: como eles podem estabelecê-la sem antes trocar DVDs? Podemos supor que Alice e Bob estejam em extremidades opostas de um cabo de fibra óptica pelo qual possam enviar e receber pulsos de luz. Porém, uma intrusa atrevida chamada Trudy pode cortar a fibra e criar um grampo ativo. Trudy pode ler todos os bits em ambos os sentidos. Ela também pode enviar falsas mensagens nos dois sentidos. A situação pode parecer desesperadora para Alice e Bob, mas a criptografia quântica pode trazer uma nova luz sobre o assunto.

A criptografia quântica se baseia no fato de que a luz se propaga em pequenos pacotes chamados **fótons**, que apresentam algumas propriedades peculiares. Além disso, a luz pode ser polarizada ao passar por um filtro de polarização, um fato bem conhecido pelos usuários de óculos de sol e pelos fotógrafos. Se um feixe de luz (isto é, um fluxo de fótons) passar por um filtro de polarização, todos os fótons que emergirem dele serão polarizados na direção do eixo do filtro (por exemplo, vertical). Se o feixe passar agora por um segundo filtro de polarização, a intensidade da luz que emergirá do segundo filtro será proporcional ao quadrado do cosseno do ângulo entre os eixos. Se os dois eixos forem perpendiculares, nenhum fóton passará pelo filtro. A orientação absoluta dos dois filtros não importa; só interessa o ângulo entre seus eixos.

Para gerar uma chave única, Alice precisa de dois conjuntos de filtros de polarização. O primeiro conjunto consiste em um filtro vertical e um filtro horizontal. Essa escolha é chamada **base retilínea**. Uma base é apenas um sistema de coordenadas. O segundo conjunto de filtros é idêntico, exceto por estar deslocado 45 graus, de forma que um filtro abrange desde o canto inferior esquerdo até o canto superior direito, e o outro abrange desde o canto superior esquerdo até o canto inferior direito. Essa escolha é chamada **base diagonal**. Desse modo, Alice tem duas bases, que ela pode inserir rapidamente em seu feixe, à vontade. Na realidade, ela não tem quatro filtros separados, mas um cristal, cuja polarização pode ser trocada eletricamente para qualquer das quatro direções permitidas, em alta velocidade. Bob tem o mesmo equipamento de Alice.

O fato de Alice e Bob terem cada um duas bases disponíveis é essencial para a criptografia quântica.

Para cada base, Alice atribui agora uma direção como 0 e a outra como 1. No exemplo apresentado a seguir, vamos supor que ela escolha a direção vertical como 0 e a horizontal como 1. Independentemente, ela também escolhe do canto inferior esquerdo até o canto superior direito como 0, e do canto superior esquerdo até o canto inferior direito como 1. Alice envia essas escolhas a Bob como texto simples.

Agora, Alice escolhe uma chave única, baseada, por exemplo, em um gerador de números aleatórios (um assunto por si só bastante complexo). Ela o transfere bit por bit para Bob, escolhendo uma de suas bases ao acaso para cada bit. Para enviar um bit, sua pistola de fótons emite um fóton polarizado de maneira apropriada, conforme a base que ela está usando para esse bit. Por exemplo, ela poderia escolher as bases diagonal, retilínea, retilínea, diagonal, retilínea etc. Para enviar sua chave única igual a 1001110010100110 com essas bases, ela enviaria os fótons mostrados na Figura 8.4(a). Dadas a chave única e a sequência de bases, a polarização a ser usada para cada bit é determinada de forma exclusiva. Bits enviados um fóton por vez são chamados **qubits** (bits quânticos).

Bob não sabe que base usar, e assim escolhe uma ao acaso para cada fóton que chega e simplesmente a utiliza, como mostra a Figura 8.4(b). Se escolher a base correta, ele receberá o bit correto. Se escolher a base incorreta, receberá um bit aleatório porque, se um fóton acessar um filtro polarizado a 45 graus em relação a sua própria polarização, ele saltará ao acaso para a polarização do filtro ou para uma polarização perpendicular à do filtro, com igual probabilidade. Essa propriedade dos fótons é fundamental para a mecânica quântica. Desse modo, alguns bits estão corretos e alguns são aleatórios, mas Bob não consegue distingui-los. Os resultados de Bob estão representados na Figura 8.4(c).

De que maneira Bob pode descobrir quais são as bases corretas e quais são as erradas entre as que recebeu? Ele apenas diz a Alice que base usou para cada bit em texto simples, e ela lhe diz quais são as bases corretas e quais são as erradas em texto simples, como mostra a Figura 8.4(d). A partir dessas informações, ambos podem construir uma sequência de bits com os palpites corretos, como mostra a Figura 8.4(e). Em média, essa sequência de bits terá metade do comprimento da sequência de bits original, mas, como ambas as partes o conhecem, elas poderão usá-lo como uma chave única. Tudo o que Alice tem a fazer é transmitir uma sequência de bits um pouco maior que o dobro do tamanho desejado, para que ela e Bob tenham uma chave única com o comprimento apropriado. Problema resolvido.

Mas espere um minuto. Esquecemos de Trudy. Vamos supor que ela esteja curiosa para saber o que Alice tem a dizer e corte o cabo de fibra, inserindo seus próprios detector e transmissor. Infelizmente para Trudy, ela também não sabe que base usar para cada fóton. O melhor que ela pode fazer é escolher uma base ao acaso para cada um dos fótons, como fez Bob. Um exemplo de suas escolhas é mostrado na Figura 8.4(f). Quando mais tarde Bob informar (em texto simples) que bases usou e Alice disser a ele (em texto simples) quais delas estão corretas, Trudy saberá quando acertou e quando errou. Segundo a Figura 8.4, ela acertou nos bits 0, 1, 2, 3,

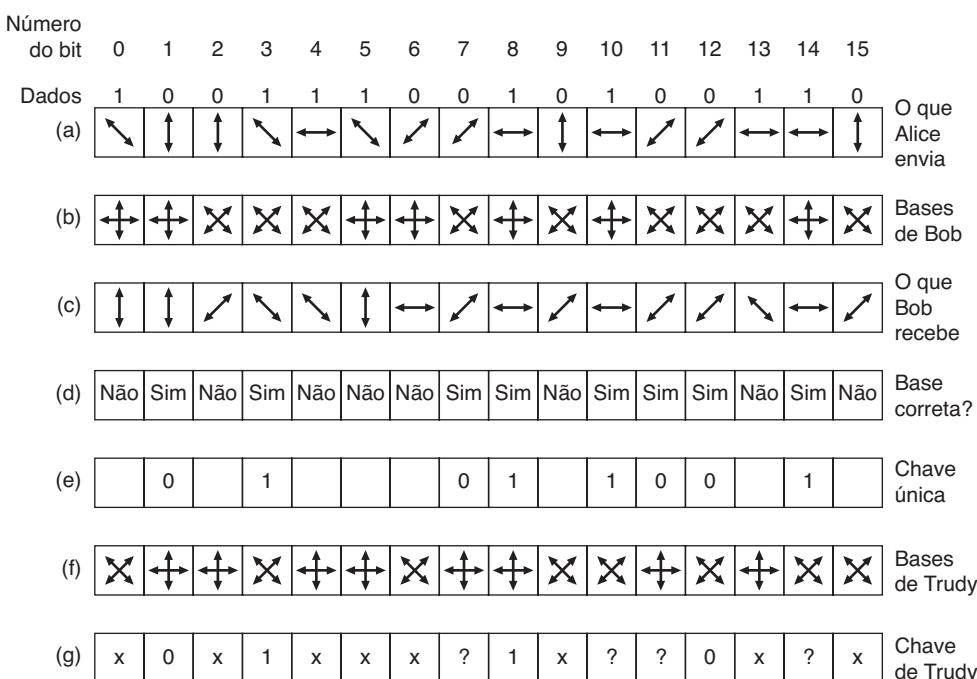



Figura 8.4 | Um exemplo de criptografia quântica.

4, 6, 8, 12 e 13. No entanto, ela sabe pela resposta de Alice na Figura 8.4(d) que só os bits 1, 3, 7, 8, 10, 11, 12 e 14 fazem parte da chave única. Em quatro desses bits (1, 3, 8 e 12), ela acertou seu palpite e capturou o bit correto. Nos outros quatro (7, 10, 11 e 14), ela errou e não sabe qual bit foi transmitido. Desse modo, Bob sabe que a chave única começa com 01011001, a partir da Figura 8.4(e), mas tudo que Trudy tem é 01?1??0?, a partir da Figura 8.4(g).

É claro que Alice e Bob estão cientes de que Trudy talvez tenha captado parte de sua chave única, e assim gostariam de reduzir as informações que ela tem. Eles podem fazer isso executando uma transformação na chave. Por exemplo, poderiam dividir a chave única em blocos de 1.024 bits e elevar ao quadrado cada uma para formar um número de 2.048 bits, usando a concatenação desses números de 2.048 bits como a chave única. Com seu conhecimento parcial da sequência de bits transmitida, Trudy não tem como gerar seu quadrado e, portanto, não tem nada. A transformação da chave única original em uma chave diferente que reduza o conhecimento de Trudy é chamada **amplificação de privacidade**. Na prática, são usadas transformações complexas em que todo bit de entrada depende de cada bit de saída em vez do quadrado do bit de entrada.

Pobre Trudy. Ela não apenas não tem nenhuma ideia de qual é a chave única, como também sua presença não é mais secreta. Afinal, ela tem de retransmitir cada bit recebido para Bob, a fim de levá-lo a pensar que está se comunicando com Alice. Porém, o melhor que pode fazer é transmitir o qubit que recebeu usando a mesma polarização que empregou para recebê-lo, e durante cerca de metade do tempo ela estará errada, provocando muitos erros na chave única de Bob.

Quando finalmente começar a transmitir dados, Alice os codificará usando um pesado código de correção antecipada de erros. Do ponto de vista de Bob, um erro de 1 bit na chave única é o mesmo que um erro de transmissão de 1 bit. De qualquer modo, ele receberá o bit errado. Se houver correção antecipada de erros suficiente, ele poderá recuperar a mensagem original apesar de todos os erros, mas poderá contar com facilidade quantos erros foram corrigidos. Se esse número for muito maior que a taxa de erros esperada do equipamento, ele saberá que Trudy grampeou a linha e poderá tomar providências (por exemplo, informando Alice de que ela deve mudar para um canal de rádio, chamar a polícia etc.). Se Trudy tivesse um meio de clonar fótons, de forma que tivesse um para inspecionar e um idêntico para enviar a Bob, poderia evitar a detecção, mas no momento não se conhece nenhum modo perfeito de clonar um fóton. No entanto, mesmo que Trudy pudesse fazê-lo, o valor da criptografia quântica para estabelecer chaves únicas não seria reduzido.

Embora a criptografia quântica opere sobre distâncias de até 60 km de fibra, o equipamento é complexo e dispendioso. Ainda assim, a ideia é promissora. Para obter mais informações sobre a criptografia quântica, consulte Mullins (2002).

8.1.5 Dois princípios fundamentais da criptografia

Ainda estudaremos muitos sistemas criptográficos nas próximas páginas, mas é importante entender dois princípios básicos subjacentes a todos eles. Preste atenção. Você correrá risco ao violá-los.

Redundância

O primeiro princípio é que todas as mensagens criptografadas devem conter alguma redundância, ou seja, informações que não são necessárias para a compreensão da mensagem. Talvez um exemplo esclareça por que isso é necessário. Considere uma empresa de encomendas postais, a Expresso Jabuti (EJ), com 60 mil produtos. Pensando ser muito eficientes, os programadores da EJ decidiram que as mensagens de encomendas devem consistir no nome do cliente com 16 bytes, seguido por um campo de dados de 3 bytes (um para a quantidade e dois para o número do produto). Os 3 últimos bytes devem ser criptografados por meio de uma chave muito longa conhecida apenas pelo cliente e pela EJ.

A princípio, essa estratégia pode parecer segura, e até certo ponto isso acontece, porque os intrusos passivos não podem descriptografar as mensagens. Infelizmente, há uma falha fatal que a torna inútil. Suponha que uma funcionária recém-demitida queira punir a EJ por tê-la despedido. Antes de sair da empresa, ela leva consigo parte da lista de clientes e passa a noite acordada criando um programa para gerar encomendas fictícias utilizando nomes de clientes verdadeiros. Como não tem a lista das chaves, ela simplesmente inclui números aleatórios nos três últimos bytes e envia centenas de encomendas para a EJ.

Quando as mensagens chegam, o computador da EJ utiliza o nome do cliente para localizar a chave e descriptografar a mensagem. Infelizmente para a EJ, quase todas as mensagens de 3 bytes são válidas; portanto, o computador começa a imprimir as instruções de entrega. Apesar de parecer estranho um cliente encomendar 837 conjuntos de balanços para crianças, ou 540 caixas de areia, para o computador, o cliente pode estar planejando abrir uma cadeia de parques de diversões por franquia. Portanto, um intruso ativo (a ex-funcionária) pode causar muitos problemas, mesmo que não seja capaz de entender as mensagens que seu computador está gerando.

Esse problema pode ser resolvido por meio da inclusão de informações redundantes em todas as mensagens. Por exemplo, se as mensagens de pedidos forem ampliadas para 12 bytes, os 9 primeiros deverão ser iguais a zero; assim, essa estratégia de ataque deixa de ser interessante, porque a ex-funcionária não é mais capaz de gerar um longo fluxo de mensagens válidas. A moral da história é que todas as mensagens devem conter informações redundantes suficientes para que os intrusos ativos sejam impedidos de transmitir dados inválidos que possam ser interpretados como uma mensagem válida.

No entanto, a inclusão de informações redundantes também facilita a quebra de mensagens por parte dos criptoanalistas. Suponha que o negócio de encomenda postal seja muito competitivo e que a principal concorrente da Expresso Jabuti, a Tartaruga Entregas, adoraria saber quantas caixas de areia a EJ está vendendo. Para isso, resolve grampear a linha telefônica da EJ. No esquema original com mensagens de 3 bytes, a criptoanálise era praticamente impossível porque, após descobrir uma chave, o criptoanalista não era capaz de dizer se a mensagem estava correta. Afinal de contas, quase todas as mensagens são tecnicamente válidas. Com o novo esquema de 12 bytes, fica mais fácil para o criptoanalista distinguir uma mensagem válida de uma inválida. Desse modo, temos:

> *Princípio criptográfico 1: as mensagens devem conter alguma redundância*

Em outras palavras, ao decifrar uma mensagem, o destinatário deve ser capaz de saber se ela é válida, simplesmente inspecionando-a e talvez executando uma computação simples. Essa redundância é necessária para impedir que intrusos ativos enviem lixo e enganem o receptor, fazendo-o descriptografar o lixo e agir sobre o 'texto simples'. No entanto, essa mesma redundância permite que os intrusos passivos entrem no sistema com maior facilidade; portanto, há uma zona de tensão nessa situação. Além disso, a redundância nunca deverá ser criada sob a forma de *n* zeros no início ou no fim de uma mensagem, pois o envio dessas mensagens a determinados algoritmos criptográficos proporciona resultados mais previsíveis, facilitando o trabalho do criptoanalista. Um polinômio de CRC é muito melhor que uma sequência de valores zero, pois o receptor pode verificá-lo facilmente, mas gerará mais trabalho para o criptoanalista. Seria muito melhor usar um hash criptográfico, conceito que exploraremos mais adiante. Por enquanto, pense nele como um CRC melhor.

Voltando à criptografia quântica por um momento, também podemos ver como a redundância desempenha um papel importante. Devido à interceptação dos fótons por Trudy, alguns bits na chave única de Bob estarão errados. Bob precisa de alguma redundância nas mensagens de entrada para descobrir os erros presentes. Uma forma muito rudimentar de redundância é repetir a mensagem duas vezes. Se as duas cópias não forem idênticas, Bob saberá que a fibra tem muito ruído, ou que alguém está interferindo na transmissão. É claro que enviar tudo duas vezes é um exagero; um código de Hamming ou de Reed-Solomon é um modo mais eficiente de realizar a detecção e correção de erros. Porém, deve ficar claro que alguma redundância é necessária para distinguir uma mensagem válida de uma mensagem inválida, em especial diante de um intruso ativo.

Atualidade

O segundo princípio criptográfico é tomar algumas medidas para assegurar que cada mensagem recebida possa ser confirmada como uma mensagem atual, isto é, enviada muito recentemente. Essa medida é necessária para impedir que intrusos ativos reutilizem mensagens antigas. Se tais medidas não fossem tomadas, nossa ex-funcionária poderia interceptar a linha telefônica da EJ e ficar simplesmente repetindo mensagens válidas já enviadas. Assim:

> *Princípio criptográfico 2: algum método é necessário para anular ataques de repetição*

Uma medida desse tipo seria incluir em cada mensagem um registro de tempo válido apenas por, digamos, 10 segundos. Em seguida, o receptor poderia manter as mensagens durante 10 segundos, a fim de comparar as mensagens recém-chegadas com as anteriores e filtrar duplicatas. As mensagens transmitidas há mais de 10 segundos poderiam ser descartadas, pois as repetições enviadas mais de 10 segundos depois da mensagem original serão rejeitadas por ser muito antigas. Outras medidas além dos registros de tempo serão discutidas mais adiante.

8.2 Algoritmos de chave simétrica

Embora a criptografia moderna utilize as mesmas ideias básicas da criptografia tradicional (transposição e substituição), sua ênfase é diferente. Tradicionalmente, os criptógrafos têm utilizado algoritmos simples. Hoje em dia, acontece o inverso: o objetivo é tornar o algoritmo de criptografia tão complexo e emaranhado que, mesmo que o criptoanalista adquira enormes volumes de texto cifrado a sua própria escolha, sem a chave ele não seja capaz de entender nada do que conseguir.

A primeira classe de algoritmos de criptografia que estudaremos neste capítulo é a dos **algoritmos de chave simétrica**, porque utilizam a mesma chave para codificação e decodificação. A Figura 8.1 ilustra o uso de um algoritmo de chave simétrica. Em particular, vamos nos concentrar nos **blocos de cifras**, que obtêm um bloco de *n* bits de texto simples como entrada e o transformam usando a chave em um bloco de *n* bits de texto cifrado.

Os algoritmos criptográficos podem ser implementados em hardware (para obter velocidade) ou em software (para obter flexibilidade). Embora a maior parte de nosso tratamento esteja relacionado aos algoritmos e protocolos, que são independentes da implementação real, algumas palavras sobre a construção de hardware criptográfico podem ser interessantes. As transposições e substituições podem ser implementadas com circuitos elétricos simples. A Figura 8.5(a) mostra um dispositivo, conhecido como **caixa P** (onde P significa permutação), usado para efetuar uma transposição em uma entrada de 8 bits. Se os 8 bits forem

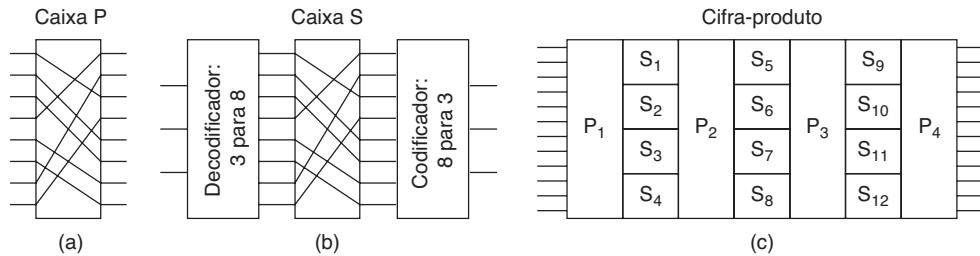



Figura 8.5 | Elementos básicos das cifras-produto. (a) Caixa P. (b) Caixa S. (c) Produto.

designados de cima para baixo como 01234567, a saída dessa caixa P específica será 36071245. Com uma conexão interna adequada, pode-se criar uma caixa P para executar qualquer transposição praticamente na velocidade da luz, pois nenhuma computação é envolvida, apenas a propagação de sinais. Esse projeto segue o princípio de Kerckhoff: o atacante sabe que o método geral é permutar os bits. O que ele não sabe é qual bit fica em cada posição.

As substituições são realizadas por **caixas S**, como mostra a Figura 8.5(b). Nesse exemplo, é introduzido um texto simples de 3 bits, e a saída é um texto cifrado de 3 bits. A entrada de 3 bits seleciona uma das oito linhas de saída do primeiro estágio e a define como 1; todas as outras são iguais a 0. O segundo estágio é uma caixa P. O terceiro estágio codifica a linha selecionada novamente em binário. Com a conexão interna mostrada, se os oito números octais 01234567 fossem introduzidos um após o outro, a sequência de saída seria 24506713. Em outras palavras, 0 foi substituído por 2, 1 foi substituído por 4 etc. Mais uma vez, com a conexão apropriada da caixa P dentro da caixa S, qualquer substituição pode ser realizada. Além disso, tal dispositivo pode ser construído em hardware e consegue alcançar grande velocidade, pois os codificadores e os decodificadores têm apenas um ou dois atrasos nas interfaces de entrada e saída (abaixo de um nanossegundo) e o tempo de propagação pela caixa P pode ser menor que 1 picossegundo.

A capacidade real desses elementos básicos se torna aparente quando dispomos uma série inteira de caixas em cascata para formar uma **cifra-produto**, como mostra a Figura 8.5(c). Nesse exemplo, 12 linhas de entrada são transpostas (isto é, permutadas) pelo primeiro estágio (P_1). No segundo estágio, a entrada é dividida em quatro grupos de 3 bits, sendo que cada um deles é substituído de forma independente dos outros (S_1 a S_4). Esse arranjo mostra um método de aproximar uma caixa S maior a partir de várias caixas S menores. Ele é útil porque caixas S menores são práticas para a implementação pelo hardware (por exemplo, uma caixa S de 8 bits pode ser observada como uma tabela de pesquisa de 256 entradas), mas caixas S grandes se tornam difíceis de criar (por exemplo, uma caixa S de 12 bits, no mínimo, precisaria de $2^{12} = 4.096$ fios cruzados em seu estágio intermediário). Apesar de ser menos genérico, esse método ainda é poderoso. Através da inclusão de um número de estágios suficientemente grande na cifra-produto, a saída pode ser transformada em uma função excessivamente complicada da entrada.

As cifras-produto que operam sobre entradas de *k* bits para produzir saídas de *k* bits são muito comuns. Em geral, o valor de *k* varia de 64 a 256. Uma implementação de hardware normalmente tem pelo menos 18 estágios físicos, em vez de apenas sete, como na Figura 8.5(c). Uma implementação de software é programada como um loop com pelo menos 8 iterações, cada uma executando substituições semelhantes às de caixas S em sub-blocos do bloco de dados de 64 bits a 256 bits, seguidas por uma permutação que mistura as saídas das caixas S. Com frequência, existe uma permutação especial no início e também uma no fim. Na literatura, as repetições são chamadas **rodadas**.

8.2.1 DES – Data Encryption Standard

Em janeiro de 1977, o governo dos Estados Unidos adotou uma cifra-produto desenvolvida pela IBM como seu padrão oficial para informações não confidenciais. A cifra, o padrão de criptografia de dados, ou **DES** (**Data Encryption Standard**), foi amplamente adotada pelo setor de informática para uso em produtos de segurança. Em sua forma original, ela já não é mais segura; no entanto, em uma forma modificada ela ainda é útil. Agora, vamos explicar como o DES funciona.

Na Figura 8.6(a), mostramos um esboço do DES. O texto simples é criptografado em blocos de 64 bits, produzindo 64 bits de texto cifrado. O algoritmo, parametrizado por uma chave de 56 bits, tem 19 estágios distintos. O primeiro deles é uma transposição independente da chave no texto simples de 64 bits. O último estágio é exatamente o inverso dessa transposição. O penúltimo estágio troca os 32 bits mais à esquerda pelos 32 bits mais à direita. Os 16 estágios restantes são funcionalmente idênticos, mas são parametrizados por diferentes funções da chave. O algoritmo foi projetado para permitir que a decodificação fosse feita com a mesma chave da codificação, uma propriedade necessária em qualquer algoritmo de chave simétrica. As etapas são executadas na ordem inversa.

A operação desses estágios intermediários é ilustrada na Figura 8.6(b). Cada estágio utiliza duas entradas de 32 bits e

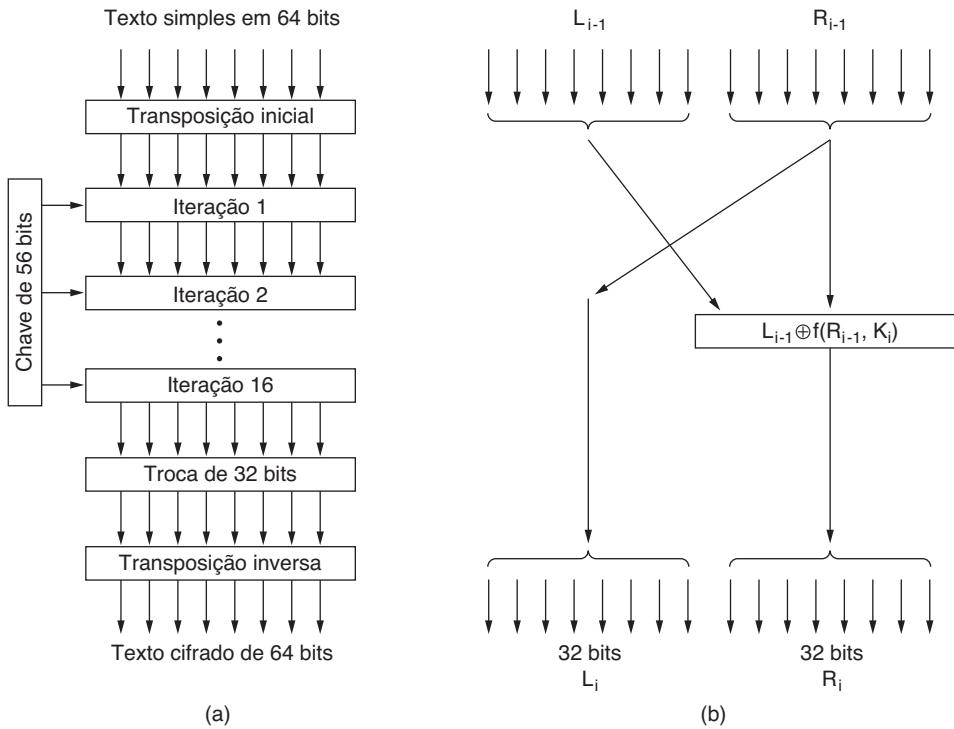



Figura 8.6 | O Data Encryption Standard. (a) Esboço geral. (b) Detalhe de uma iteração. O sinal de adição dentro do círculo significa OU exclusivo (XOR).

produz duas saídas de 32 bits. A saída da esquerda é apenas uma cópia da saída da direita. A saída da direita é formada pelo resultado do OU exclusivo bit a bit aplicado à entrada da esquerda e a uma função de entrada da direita com a chave desse estágio, K_i. Toda a complexidade reside nessa função.

A função consiste em quatro etapas, executadas em sequência. Primeiro, um número de 48 bits, E, é construído através da expansão do R_{i-1} de 32 bits, de acordo com uma regra fixa de transposição e duplicação. Em segundo lugar, E e K_i são submetidos a uma operação XOR. Em seguida, essa saída é dividida em oito grupos de 6 bits, sendo cada um deles entregue a uma caixa S diferente. Cada uma das 64 entradas possíveis para uma caixa S é mapeada em uma saída de 4 bits. Por fim, esses 8×4 bits passam por uma caixa P.

Em cada uma das 16 iterações, é utilizada uma chave diferente. Antes de se iniciar o algoritmo, é aplicada à chave uma transposição de 56 bits. Antes de cada iteração, a chave é particionada em duas unidades de 28 bits, sendo cada uma delas girada à esquerda um número de bits que depende do número da iteração. K_i é derivada dessa chave girada, pela aplicação de mais uma transposição de 56 bits sobre ela. Em cada rodada, um novo subconjunto de 48 bits dos 56 bits é extraído e permutado.

Uma técnica às vezes utilizada para tornar o DES mais forte é chamada **clareamento**. Ela consiste em operação XOR entre uma chave aleatória de 64 bits e cada bloco de texto simples, antes de sua entrega ao DES, e depois em uma operação XOR entre uma segunda chave de 64 bits e o texto cifrado resultante, antes de sua transmissão. O clareamento pode ser removido com facilidade pela execução das operações inversas (se o receptor tiver as duas chaves de clareamento). Como essa técnica acrescenta efetivamente mais bits ao tamanho da chave, ela torna uma pesquisa exaustiva do espaço de chaves muito mais demorada. Observe que a mesma chave de clareamento é utilizada para cada bloco (isto é, só existe uma chave de clareamento).

O DES esteve envolvido em controvérsias desde seu lançamento. Ele se baseia em uma cifra desenvolvida e patenteada pela IBM, cujo nome é Lucifer. A diferença é que a cifra da IBM utilizava uma chave de 128 bits, em vez de uma chave de 56 bits. Quando quis padronizar o uso de uma cifra para informações não confidenciais, o governo dos Estados Unidos ‘convidou’ a IBM para ‘discutir’ o problema com a NSA, o órgão do governo especializado em decifrar códigos, que é o maior empregador de matemáticos e criptólogos do mundo. A NSA é tão secreta que no setor existe a seguinte brincadeira com seu nome:

P: O que significa NSA?

R: No Such Agency (em português: não existe tal agência).

Na verdade, NSA significa National Security Agency (Agência de Segurança Nacional).

Após essas discussões, a IBM reduziu a chave de 128 para 56 bits e decidiu manter em segredo o processo se-

gundo o qual o DES foi projetado. Muita gente suspeitou de que o tamanho da chave tinha sido reduzido para garantir que a NSA pudesse decifrar o DES, mas nenhuma organização com um orçamento menor foi capaz de fazê-lo. Supostamente, o motivo para manter o projeto em segredo foi ocultar uma porta dos fundos que pudesse facilitar ainda mais a decifração do DES por parte da NSA. Quando um funcionário da NSA disse discretamente para o IEEE (Institute of Electrical and Electronic Engineers) cancelar uma conferência sobre criptografia, as pessoas ficaram ainda mais embaraçadas com a situação. A NSA negou tudo.

Em 1977, dois pesquisadores de criptografia de Stanford, Diffie e Hellman (1977), projetaram uma máquina para decifrar o DES e estimaram que ela poderia ser montada por um custo de 20 milhões de dólares. Com base em um pequeno trecho de texto simples e no texto cifrado correspondente, tal máquina poderia descobrir a chave através de uma pesquisa exaustiva do espaço de chaves de 2^{56} entradas em menos de um dia. Atualmente, já se pode brincar. A máquina existe, está à venda e custa menos de US$ 10.000 para ser fabricada (Kumar et al., 2006).

DES triplo

No início de 1979, a IBM percebeu que o tamanho da chave do DES era muito pequeno e criou uma forma de aumentá-lo usando a criptografia tripla (Tuchman, 1979). O método escolhido, que desde então foi incorporado ao padrão internacional 8732, está ilustrado na Figura 8.7. Nesse caso, são usados três estágios e duas chaves. No primeiro estágio, o texto simples é criptografado com K_1 da maneira usual do DES. No segundo estágio, o DES é executado no modo de descriptografia, com o uso de K_2 como chave. Por fim, outra criptografia DES é feita com K_1.

Esse projeto levanta duas questões. Primeiro, por que são utilizadas apenas duas chaves em vez de três? Segundo, por que foi usado **EDE** (**Encrypt Decrypt Encrypt**) em vez de **EEE** (**Encrypt Encrypt Encrypt**)? São utilizadas duas chaves porque até mesmo os criptógrafos mais paranoicos concordam que 112 bits serão suficientes para aplicações comerciais durante um bom tempo. (E, entre os criptógrafos, a paranoia é considerada um recurso, não um bug.) O uso de 168 bits só criaria overhead desnecessário para gerenciar e transportar outra chave, com pouco ganho real.

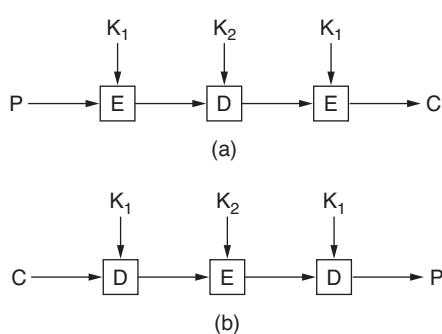



Figura 8.7 | Criptografia tripla usando o DES. (b) Descriptografia.

O motivo para criptografar, descriptografar e criptografar mais uma vez é a compatibilidade com os sistemas DES de chave única existentes. Tanto as funções de criptografia quanto as de descriptografia são mapeamentos entre conjuntos de números de 64 bits. Do ponto de vista da criptografia, os dois mapeamentos são igualmente fortes. No entanto, ao usar EDE em vez de EEE, um computador que utiliza a criptografia tripla pode se comunicar com outro que utiliza a criptografia simples apenas definindo $K_1 = K_2$. Essa propriedade permite que a criptografia tripla seja implantada gradualmente, o que não interessa aos criptógrafos acadêmicos, mas é de grande importância para a IBM e seus clientes.

8.2.2 AES – Advanced Encryption Standard

À medida que a vida útil do DES começou a se aproximar do fim, mesmo com o DES triplo, o **NIST** (**National Institute of Standards and Technology**), o órgão do departamento de comércio dos Estados Unidos encarregado de aprovar padrões, decidiu que o governo precisava de um novo padrão criptográfico para uso não confidencial. O NIST estava ciente de toda a controvérsia que cercava o DES e sabia muito bem que, se anunciasse um novo padrão, todas as pessoas com algum conhecimento sobre criptografia logo concluiriam que a NSA havia criado uma porta dos fundos no DES, e assim poderia ler tudo que fosse criptografado com ele. Em tais condições, talvez ninguém utilizasse o padrão e seria mais provável que ele desaparecesse.

Dessa forma, o NIST adotou uma estratégia diferente e surpreendente para um órgão do governo: patrocinou uma competição de criptografia. Em janeiro de 1997, pesquisadores do mundo inteiro foram convidados a submeter propostas para um novo padrão, a ser chamado **AES** (**Advanced Encryption Standard**). As regras dessa competição eram:

1. o algoritmo teria de ser uma cifra de bloco simétrica;
2. todo o projeto teria de ser público;
3. deveriam ser admitidos tamanhos de chaves iguais a 128, 192 e 256 bits;
4. teriam de ser possíveis implementações de software e de hardware;
5. o algoritmo teria de ser público ou licenciado em termos não discriminatórios.

Foram feitas quinze propostas sérias e organizadas conferências públicas nas quais essas propostas eram apresentadas, e os participantes encorajados ativamente a encontrar falhas em todas elas. Em agosto de 1998, o NIST selecionou cinco finalistas, baseado principalmente em seus requisitos de segurança, eficiência, simplicidade, flexibilidade e memória (importantes para sistemas embarcados).

Foram realizadas outras conferências e mais tentativas de encontrar falhas nos algoritmos.

Em outubro de 2000, o NIST anunciou que tinha selecionado o Rijndael, de Joan Daemen e Vincent Rijmen. O nome é derivado do sobrenome dos autores: Rijmen + Daemen. Em novembro de 2001, o Rijndael se tornou o padrão AES do governo dos Estados Unidos, publicado como FIPS (Federal Information Processing Standard) 197. Por causa da extraordinária abertura da competição, das propriedades técnicas do Rijndael e do fato de a equipe premiada consistir em dois jovens criptógrafos belgas (com pouca probabilidade de ter criado uma porta dos fundos só para agradar a NSA), o Rijndael se tornou o padrão criptográfico dominante no mundo. A criptografia e a descriptografia AES fazem parte do conjunto de instruções de alguns microprocessadores (por exemplo, Intel).

O Rijndael admite tamanhos de chaves e tamanhos de blocos desde 128 bits até 256 bits em intervalos de 32 bits. O comprimento da chave e o do bloco podem ser escolhidos independentemente. Porém, o AES especifica que o tamanho do bloco deve ser 128 bits e o comprimento da chave deve ser 128, 192 ou 256 bits. É pouco provável que alguém utilize chaves de 192 bits; assim, de fato, o AES tem duas variantes: um bloco de 128 bits com uma chave de 128 bits e um bloco de 128 bits com uma chave de 256 bits.

Em nosso tratamento do algoritmo, examinaremos apenas o caso de 128/128, porque é provável que esse se torne a norma comercial. Uma chave de 128 bits oferece um espaço de chaves de $2^{128} \approx 3 \times 10^{38}$ chaves. Ainda que a NSA consiga construir uma máquina com 1 bilhão de processadores paralelos, cada um capaz de avaliar uma chave por picossegundo, tal máquina levaria cerca de 10^{10} anos para pesquisar o espaço de chaves. Nessa época, o Sol já terá explodido e as pessoas que sobreviverem terão de ler os resultados à luz de velas.

Rijndael

Sob uma perspectiva matemática, o Rijndael se baseia na teoria de campo de Galois, o que proporciona ao algoritmo algumas propriedades de segurança demonstráveis. Porém, ele também pode ser visto como código em C, sem a necessidade de entrarmos nos detalhes matemáticos.

Como o DES, o Rijndael utiliza substituição e permutações, e também emprega várias rodadas. O número de rodadas depende do tamanho da chave e do tamanho do bloco, sendo 10 para chaves de 128 bits com blocos de 128 bits, passando para 14 no caso da maior chave ou do maior bloco. No entanto, diferentemente do DES, todas as operações envolvem bytes inteiros, a fim de permitir implementações eficientes, tanto em hardware quanto em software. Um esboço do código é apresentado no Quadro 8.1. Observe que esse código é para fins de ilustração. Boas implementações do código de segurança seguirão práticas adicionais, como zerar a memória confidencial depois que ela tiver sido usada. Veja, por exemplo, Ferguson et al. (2010).

```
#define LENGTH 16                                      /* número de bytes no bloco de dados ou chave */
#define NROWS 4                                        /* número de linhas no estado */
#define NCOLS 4                                        /* número de colunas no estado */
#define ROUNDS 10                                      /* número de iterações */
typedef unsigned char byte;                            /* inteiro de 8 bits sem sinal */
rijndael(byte plaintext[LENGTH], byte ciphertext[LENGTH], byte key[LENGTH])
{
  int r;                                               /* índice do loop */
  byte state[NROWS][NCOLS];                            /* estado atual */
  struct {byte k[NROWS][NCOLS];} rk[ROUNDS + 1];       /* chaves da rodada */
  expand_key(key, rk);                                 /* constrói as chaves da rodada */
  copy_plaintext_to_state(state, plaintext);           /* inicia estado atual */
  xor_roundkey_into_state(state, rk[0]);               /* XOR da chave no estado */
  for (r = 1; r <= ROUNDS; r++) {
      substitute(state);                               /* aplica caixa S a cada byte */
      rotate_rows(state);                              /* gira linha i por i bytes */
      if (r < ROUNDS) mix_columns(state);              /* função embaralha */
      xor_roundkey_into_state(state, rk[r]);           /* XOR da chave no estado */
  }
  copy_state_to_ciphertext(ciphertext, state);         /* retorna resultado */
}
```

Quadro 8.1 | Um esboço do Rijndael em C.

A função *rijndael* tem três parâmetros: *plaintext*, um array de 16 bytes contendo os dados de entrada, *ciphertext*, um array de 16 bytes no qual será retornada a saída cifrada, e *key*, a chave de 16 bytes. Durante o cálculo, o estado atual dos dados é mantido no array de bytes *state*, cujo tamanho é *NROWS* × *NCOLS*. No caso de blocos de 128 bits, esse array tem 4 × 4 bytes. Em 16 bytes, é possível armazenar todo o bloco de dados de 128 bits.

O array *state* é iniciado com o texto simples e modificado em cada etapa da computação. Em algumas etapas, é executada a substituição byte a byte. Em outras etapas, os bytes são permutados dentro do array. Outras transformações também são usadas. No final, o conteúdo do array *state* é retornado como texto cifrado.

O código começa expandindo a chave em 11 arrays do mesmo tamanho do estado. Eles são armazenados em *rk*, um array de structs, cada um contendo um array de estado. Um desses arrays será utilizado no início do cálculo, e os outros 10 serão usados durante as 10 rodadas, um por rodada. O cálculo das chaves de rodadas a partir da chave de criptografia é muito complicado para ser examinado aqui. Basta saber que as chaves de rodadas são produzidas pela rotação e pela operação XOR repetida em vários grupos de bits da chave. Para ver todos os detalhes, consulte (Daemen e Rijmen, 2002).

A próxima etapa é copiar o texto simples no array *state* de forma que ele possa ser processado durante as rodadas. O texto é copiado em ordem de colunas, com os quatro primeiros bytes na coluna 0, os quatro bytes seguintes na coluna 1 e assim por diante. Tanto as colunas quanto as linhas são numeradas a partir de 0, embora as rodadas se iniciem em 1. Essa configuração inicial dos 12 arrays de bytes com tamanho 4 × 4 é ilustrada na Figura 8.8.

Há mais uma etapa antes do início da principal computação: *rk*[0] é submetido a uma operação XOR em *state*, byte por byte. Em outras palavras, cada um dos 16 bytes em *state* é substituído pelo XOR do próprio valor com o byte correspondente em *rk*[0].

Agora chegou a hora da atração principal. O loop executa dez iterações, uma por rodada, transformando *state* em cada repetição. O conteúdo de cada rodada consiste em quatro etapas. A etapa 1 efetua uma substituição byte a byte em *state*. Por sua vez, cada byte é usado como um índice para uma caixa S, a fim de substituir seu valor pelo conteúdo dessa entrada da caixa S. Essa etapa é uma cifra de substituição monoalfabética. Diferentemente do DES, que tem diversas caixas S, o Rijndael tem apenas uma caixa S.

A etapa 2 gira cada uma das quatro linhas para a esquerda. A linha 0 é girada 0 bytes (isto é, não se altera), a linha 1 é girada 1 byte, a linha 2 é girada 2 bytes e a linha 3 é girada 3 bytes. Essa etapa difunde o conteúdo dos dados atuais em torno do bloco, de modo semelhante às permutações da Figura 8.5.

A etapa 3 embaralha cada coluna independentemente das outras. O embaralhamento é realizado com o uso da multiplicação de matrizes, na qual a nova coluna é o produto da coluna antiga por uma matriz constante, sendo a multiplicação feita com o campo finito de Galois, $GF(2^8)$. Embora isso possa parecer complicado, existe um algoritmo que permite calcular cada elemento da nova coluna usando duas pesquisas em tabelas e três operações XOR (Daemen e Rijmen, 2002, Apêndice E).

Por fim, a etapa 4 efetua o XOR da chave correspondente a essa rodada no array *state*.

Tendo em vista que cada etapa é reversível, a decodificação pode ser feita simplesmente executando-se o algoritmo no sentido inverso. Porém, também existe um artifício pelo qual a decodificação pode ser realizada executando-se o algoritmo de criptografia com a utilização de tabelas diferentes.

O algoritmo foi projetado não só por segurança, mas também para aumentar a velocidade. Uma boa implementação de software em uma máquina de 2 GHz deve ser capaz de alcançar uma taxa de criptografia de 700 Mbps, que é rápida o suficiente para codificar mais de cem vídeos MPEG-2 em tempo real. As implementações de hardware são ainda mais rápidas.

8.2.3 Modos de cifra

Apesar de toda essa complexidade, o AES (ou o DES, ou ainda qualquer cifra de bloco) é basicamente uma cifra

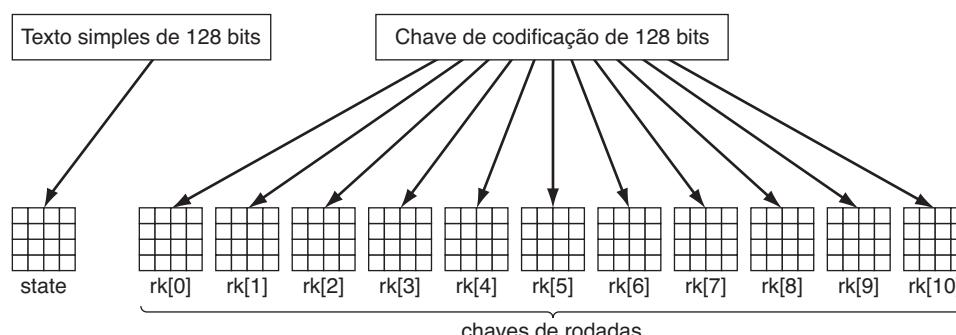



Figura 8.8 | Criação dos arrays *state* e *rk*.

de substituição monoalfabética que utiliza caracteres grandes (caracteres de 128 bits para AES, e caracteres de 64 bits para DES). Sempre que o mesmo bloco de texto simples chega ao front end, o mesmo bloco de texto cifrado sai pelo back-end. Se você codificar o texto simples *abcdefgh* 100 vezes com a mesma chave DES, obterá o mesmo texto cifrado cem vezes. Um intruso pode explorar essa propriedade para ajudar a subverter a cifra.

O modo Electronic Code Book

Para ver como essa propriedade das cifras de substituição monoalfabética pode ser usada para anular parcialmente a cifra, usaremos o DES (triplo), por ser mais fácil representar blocos de 64 que de 128 bits, mas o AES tem exatamente o mesmo problema. A maneira direta de usar o DES para codificar um longo fragmento de texto simples é dividi-lo em blocos consecutivos de 8 bytes (64 bits) e codificá-los uns após outros com a mesma chave. O último fragmento de texto simples é completado até 64 bits, se necessário. Essa técnica é conhecida como **modo ECB** (**Electronic Code Book**), em analogia aos antigos livros de código em que cada palavra de texto simples era listada, seguida por seu texto cifrado (em geral, um número decimal de cinco dígitos).

Na Figura 8.9, temos o início de um arquivo de computador com a lista das gratificações anuais que uma empresa decidiu oferecer a seus funcionários. Esse arquivo consiste em registros de 32 bytes consecutivos, um por funcionário, no formato mostrado: 16 bytes para o nome, 8 bytes para o cargo e 8 bytes para a gratificação. Cada um dos dezesseis blocos de 8 bytes (numerados de 0 até 15) é codificado pelo DES (triplo).

Leslie acabou de brigar com o chefe e sabe que não deve esperar uma grande gratificação. Ao contrário, Kim é a favorita do chefe e todo mundo sabe disso. Leslie pode acessar o arquivo após a codificação, mas antes de ser enviado ao banco. Ela pode retificar essa situação injusta, tendo apenas o arquivo criptografado?

Não há problema. Leslie só precisa fazer uma cópia do 12º bloco do texto cifrado (que contém a gratificação de Kim) e usá-lo para substituir o quarto bloco do texto cifrado (que contém a própria gratificação). Mesmo sem saber o que contém o 12º bloco, Leslie pode esperar ter um Natal muito mais feliz naquele ano. (Copiar o oitavo bloco também é uma possibilidade, mas o risco de ser descoberto é maior; além disso, Leslie não é uma pessoa gananciosa.)

Modo de encadeamento de blocos de cifras

Para contrariar esse tipo de ataque, todos os blocos de cifras podem ser encadeados de várias maneiras, a fim de que a substituição de um bloco como a que Leslie fez transforme o texto simples decodificado em lixo, a partir do bloco substituído. Uma forma de encadeamento é o **encadeamento de blocos de cifras**. Nesse método, mostrado na Figura 8.10, cada bloco de texto simples é submetido a uma operação XOR com o bloco de texto cifrado anterior, antes de ser codificado. Como consequência, o mesmo bloco de texto simples não é mais mapeado

Figura 8.9 | O texto simples de um arquivo codificado como 16 blocos DES.

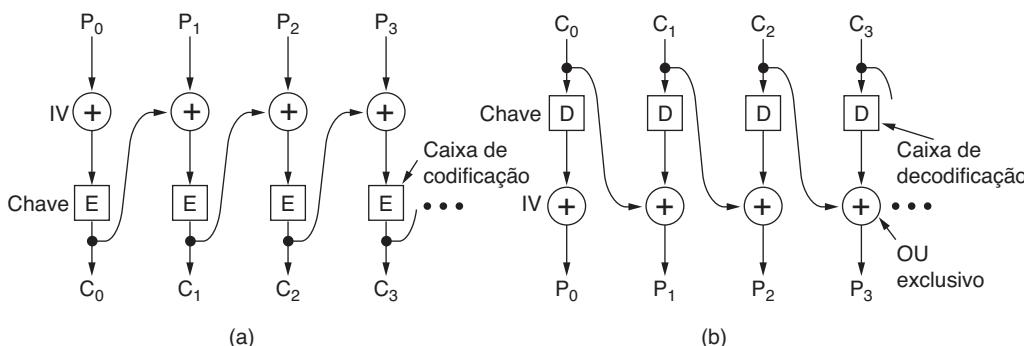



Figura 8.10 | Encadeamento de blocos de cifras. (a) Codificação. (b) Decodificação.

para o mesmo bloco de texto cifrado, e a criptografia não é mais uma grande cifra de substituição monoalfabética. O primeiro bloco é submetido a uma operação XOR com um vetor de referência, ou **IV** (**Initialization Vector**), escolhido ao acaso, que é transmitido (em texto simples) juntamente com o texto cifrado.

Podemos ver como funciona o modo de encadeamento de blocos de cifras examinando o exemplo da Figura 8.10. Começamos com o cálculo $C_0 = E(P_0 \text{ XOR } IV)$. Em seguida, calculamos $C_1 = E(P_1 \text{ XOR } C_0)$ e assim por diante. A decodificação também utiliza XOR para inverter o processo, com $P_0 = IV \text{ XOR } D(C_0)$ e assim por diante. Observe que a criptografia do bloco i é uma função de todo o texto simples contido nos blocos 0 a $i - 1$, e assim o mesmo texto simples gera um texto cifrado diferente, dependendo de onde ele ocorre. Uma transformação do tipo que Leslie fez resultará em texto sem sentido para dois blocos a partir do campo de gratificação de Leslie. Para um funcionário de segurança astuto, essa peculiaridade pode sugerir onde iniciar a investigação legal.

O encadeamento de blocos de cifras também tem uma vantagem: o mesmo bloco de texto simples não resultará no mesmo bloco de texto cifrado. Assim, a criptoanálise será difícil. De fato, essa é a principal razão de seu uso.

Modo de feedback de cifra

No entanto, o encadeamento de blocos de cifras tem a desvantagem de exigir a chegada de um bloco de 64 bits inteiro para poder iniciar a decodificação. No caso da codificação byte a byte, é usado o **modo de feedback de cifra**, empregando o DES (triplo), como mostra a Figura 8.11. Para o AES, a ideia é exatamente a mesma, sendo usado um registrador de deslocamento de 128 bits. Na figura, o estado da máquina de criptografia é mostrado após os bytes 0 a 9 terem sido codificados e enviados. Ao chegar o byte 10 do texto simples, conforme ilustra a Figura 8.11(a), o algoritmo DES opera sobre o registrador de deslocamento de 64 bits para gerar um texto cifrado de 64 bits. O byte mais à esquerda desse texto cifrado é extraído e submetido a uma operação XOR com P_{10}. Depois, esse byte é encaminhado à linha de transmissão. Além disso, o registrador de deslocamento (shift register) é deslocado 8 bits à esquerda, fazendo C_2 ficar fora da extremidade esquerda, e C_{10} é inserido na posição que acabou de ficar vaga na extremidade direita, logo depois de C_9.

Observe que o conteúdo do registrador de deslocamento depende do histórico anterior do texto simples; assim, um padrão que se repetir várias vezes no texto simples será criptografado de maneira diferente do texto cifrado a cada vez. Como ocorre no encadeamento de blocos de cifras, é necessário um vetor de referência para iniciar o processo.

A decodificação com o modo de feedback de cifra funciona exatamente como a codificação. Em particular, o conteúdo do registrador de deslocamento é *codificado*, e não *decodificado*, e assim o byte selecionado que é submetido à operação XOR com C_{10} para obter P_{10} é o mesmo que sofreu a operação XOR com P_{10} para gerar C_{10} na primeira vez. Desde que os dois registradores de deslocamento permaneçam idênticos, a decodificação funciona corretamente. Ela é ilustrada na Figura 8.11(b).

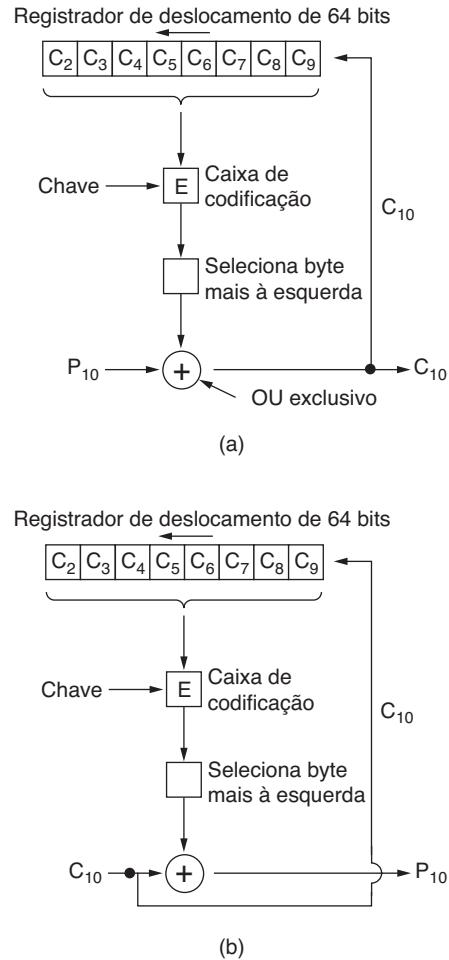



Figura 8.11 | Modo de feedback de cifra. (a) Codificação. (b) Decodificação.

Um problema apresentado pelo modo de feedback de cifra é que, se um bit do texto cifrado for invertido acidentalmente durante a transmissão, os 8 bytes decodificados enquanto o byte defeituoso estiver no registrador de deslocamento serão danificados. Depois que o byte defeituoso for empurrado para fora do registrador de deslocamento, o texto simples correto será gerado mais uma vez. Desse modo, os efeitos de um único bit invertido são em parte localizados e não danificam o restante da mensagem, mas danificam uma quantidade de bits igual à largura do registrador de deslocamento.

Modo fluxo de cifras

Apesar disso, existem aplicações em que um erro de transmissão de 1 bit alterando 64 bits de texto simples pro-

voca um impacto grande demais. Para essas aplicações, há uma quarta opção, **o modo fluxo de cifras**. Ele funciona pela codificação de um vetor de referência com uma chave para obter um bloco de saída. O bloco de saída é então codificado, usando-se a chave para obter um segundo bloco de saída. Em seguida, esse bloco é codificado para obter um terceiro bloco e assim por diante. A sequência (arbitrariamente grande) de blocos de saída, chamada **fluxo de chaves**, é tratada como uma chave única e submetida a uma operação XOR com o texto simples para obter o texto cifrado, como mostra a Figura 8.12(a). Observe que o vetor de referência só e usado na primeira etapa. Depois disso, a saída é codificada. Observe também que o fluxo de chaves é independente dos dados, portanto pode ser calculado com antecedência, se necessário, e é completamente insensível a erros de transmissão. A decodificação aparece na Figura 8.12(b).

A decodificação ocorre quando se gera o mesmo fluxo de chaves no lado receptor. Como o fluxo de chaves só depende do vetor de referência e da chave, ele não é afetado por erros de transmissão no texto cifrado. Desse modo, um erro de 1 bit no texto cifrado transmitido gera apenas um erro de 1 bit no texto simples decodificado.

É essencial nunca utilizar o mesmo par (chave, vetor de referência) duas vezes em fluxo de cifras, porque isso vai gerar o mesmo fluxo de chaves o tempo todo. O uso de um mesmo fluxo de chaves duas vezes expõe o texto cifrado a um ataque de **reutilização de fluxo de chaves**. Imagine que o bloco de texto simples P_0 seja codificado com o fluxo de chaves para obter P_0 XOR K_0. Mais tarde, um segundo bloco de texto simples Q_0 é codificado com o mesmo fluxo de chaves para obter Q_0 XOR K_0. Um intruso que capturar ambos os blocos do texto cifrado poderá simplesmente efetuar uma operação XOR dos dois juntos para obter P_0 XOR Q_0, o que eliminará a chave. Agora, o intruso tem o XOR dos dois blocos de texto simples. Se um deles for conhecido ou puder ser encontrado, o outro também poderá ser encontrado. Em todo caso, o XOR de dois fluxos de texto simples poderá ser atacado com a utilização de propriedades estatísticas da mensagem. Por exemplo, no caso de texto em inglês, o caractere mais comum no fluxo provavelmente será o XOR de dois espaços, seguido pelo XOR de espaço e da letra 'e' etc. Em resumo, equipado com o XOR de dois textos simples, o criptoanalista tem uma excelente chance de deduzi-los.

Figura 8.12 | Um fluxo de cifras. (a) Codificação. (b) Decodificação.

Modo contador

Um problema apresentado por todos os modos, com exceção do modo do livro de código eletrônico, é a impossibilidade de conseguir acesso aleatório a dados codificados. Por exemplo, suponha que um arquivo seja transmitido por uma rede e depois armazenado em disco em forma codificada. Isso poderia ser um meio razoável de operação, se o computador receptor fosse um notebook que pudesse ser roubado. Armazenar todos os arquivos críticos em forma codificada reduz muito os danos causados pelo vazamento de informações secretas na eventualidade de o computador cair em mãos erradas.

Porém, com frequência, os arquivos de disco são acessados em ordem não sequencial, especialmente arquivos de bancos de dados. No caso de um arquivo codificado pela utilização do encadeamento de blocos de cifras, o acesso a um bloco aleatório exige primeiro a decodificação de todos os blocos situados à frente dele, uma proposta dispendiosa. Por essa razão, foi criado mais um modo, o **modo contador**, como ilustra a Figura 8.13. Aqui, o texto simples não é codificado diretamente. Em vez disso, o vetor de referência somado a uma constante é codificado, e o texto cifrado resultante é submetido a um XOR com o texto simples. Aumentar o vetor de referência em uma unidade a cada novo bloco facilita a decodificação de um bloco em qualquer lugar no arquivo sem que primeiro seja preciso decodificar todos os seus predecessores.

Embora seja útil, o modo do contador tem um ponto fraco que vale a pena assinalar. Suponha que a mesma chave K seja usada novamente no futuro (com um texto simples diferente, mas com o mesmo vetor de referência) e um atacante adquira todo o texto cifrado em ambas as execuções. O fluxo de chaves é o mesmo nos dois casos, expondo a cifra a um ataque de reutilização do fluxo de chaves do tipo que vimos no caso de fluxo de cifras. O criptoanalista tem de efetuar a operação XOR dos dois textos cifrados juntos, a fim de eliminar toda a proteção criptográfica e simplesmente obter o XOR dos textos simples. Essa

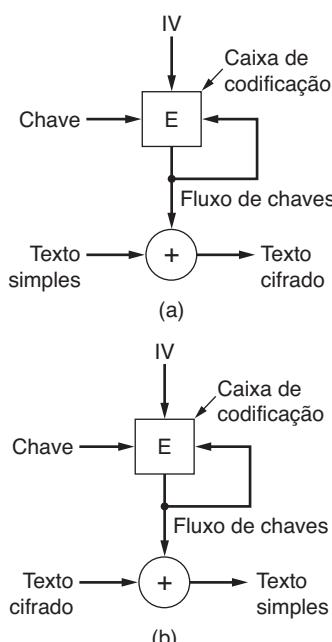
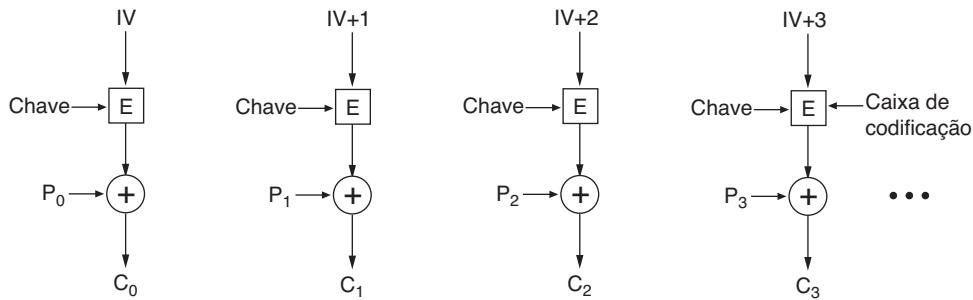



Figura 8.13 | Codificação com a utilização do modo contador.

fragilidade não significa que o modo do contador seja uma má ideia. Ela simplesmente quer dizer que tanto as chaves quanto os vetores de referência devem ser escolhidos de forma independente e ao acaso. Ainda que a mesma chave seja utilizada duas vezes por acidente, se o IV for diferente em cada utilização, o texto simples estará seguro.

8.2.4 Outras cifras

AES (Rijndael) e DES são os algoritmos criptográficos de chave simétrica mais conhecidos e as escolhas-padrão da indústria, mesmo apenas por motivos de responsabilidade. (Ninguém o culpará se você usar AES em seu produto e o AES for quebrado, mas certamente alguém o fará se você usar uma cifra fora do padrão e ela mais tarde for quebrada.) Porém, vale a pena mencionar que foram criadas várias outras cifras de chave simétrica. Algumas delas estão incorporadas a vários produtos. A Tabela 8.2 mostra algumas entre as cifras de chave simétrica mais comuns. É possível usar combinações dessas cifras, por exemplo, AES sobre Twofish, de modo que ambas precisem ser quebradas para recuperar os dados.

8.2.5 Criptoanálise

Antes de deixar o assunto de criptografia de chave simétrica, vale a pena mencionar pelo menos quatro desenvolvimentos em criptoanálise. O primeiro desenvolvimento é a **criptoanálise diferencial** (Biham e Shamir, 1997). Essa técnica pode ser usada para atacar qualquer cifra de bloco. Ela funciona a partir de um par de blocos e texto simples que diferem apenas por um pequeno número de bits e pela observação cuidadosa do que acontece em cada iteração interna à medida que a codificação prossegue. Em muitos casos, alguns padrões de bits são muito mais comuns que outros, e essa observação pode levar a ataques probabilísticos.

O segundo desenvolvimento que vale a pena notar é a **criptoanálise linear** (Matsui, 1994). Ela pode quebrar o DES com apenas 2^{43} textos simples conhecidos. Essa técnica funciona efetuando o XOR de certos bits no texto simples e no texto cifrado, e examinando o resultado. Quando isso é feito repetidamente, metade dos bits deve ter o valor 0 e metade deve ter o valor 1. Porém, com frequência, as cifras introduzem uma inclinação em um sentido ou no outro, e essa inclinação, embora pequena, pode ser explorada para reduzir o fator de trabalho. Para ver os detalhes, consulte o ensaio de Matsui.

O terceiro desenvolvimento é o **uso da análise do consumo de energia elétrica** para encontrar chaves secretas. Em geral, os computadores utilizam 3 volts para representar um bit 1 e 0 volts para representar um bit 0. Desse modo, o processamento de um bit 1 exige mais energia elétrica que o processamento de um 0. Se um algoritmo

Cifra	Autor	Comprimento da chave	Comentários
DES	IBM	56 bits	Muito fraco para usar agora
RC4	Ronald Rivest	1 a 2.048 bits	Atenção: algumas chaves são fracas
RC5	Ronald Rivest	128 a 256 bits	Bom, mas patenteado
AES (Rijndael)	Daemen e Rijmen	128 a 256 bits	Melhor escolha
Serpent	Anderson, Biham, Knudsen	128 a 256 bits	Muito forte
DES triplo	IBM	168 bits	Bom, mas está ficando ultrapassado
Twofish	Bruce Schneier	128 a 256 bits	Muito forte; amplamente utilizado

Tabela 8.2 | Alguns algoritmos criptográficos de chave simétrica comuns.

criptográfico consistir em um loop no qual os bits da chave são processados em ordem, um atacante que substituir o clock principal de n GHz por um clock lento (por exemplo, 100 Hz) e prender pinças dentadas (pinças jacaré) nos pinos de energia da CPU e de terra poderá monitorar com precisão a energia consumida por cada instrução de máquina. A partir desses dados, será surpreendentemente fácil deduzir a chave. Esse tipo de criptoanálise só pode ser anulado por codificação cuidadosa do algoritmo em linguagem assembly para ter certeza de que o consumo de energia será independente da chave e também independente de todas as chaves de rodadas individuais.

O quarto desenvolvimento é a **análise de sincronismo**. Os algoritmos criptográficos estão repletos de instruções if que testam bits nas chaves de rodadas. Se as partes then e else demorarem períodos de tempo diferentes, tornando mais lento o clock e verificando quanto tempo demoram diversas etapas, talvez seja possível deduzir as chaves de rodadas. Uma vez que todas as chaves de rodadas sejam conhecidas, em geral será possível calcular a chave original. A análise da energia e da sincronização também pode ser empregada simultaneamente para facilitar o trabalho. Embora a análise da energia e da sincronização possa parecer exótica, na realidade são técnicas eficientes que podem romper qualquer cifra não projetada de forma específica para resistir a elas.

8.3 Algoritmos de chave pública

O problema da distribuição de chaves sempre foi o elo mais fraco da maioria dos sistemas de criptografia. Independentemente de quanto um sistema de criptografia fosse sólido, se um intruso conseguisse roubar a chave, o sistema acabava sendo inútil. Como todos os criptólogos sempre presumem que a chave de criptografia e a chave de descriptografia são iguais (ou facilmente derivadas uma da outra) e que a chave é distribuída a todos os usuários do sistema, tinha-se a impressão de que havia um problema inerente ao sistema. As chaves tinham de ser protegidas contra roubo, mas também tinham de ser distribuídas; portanto, não podiam ser simplesmente trancadas na caixa-forte de um banco.

Em 1976, dois pesquisadores da Universidade de Stanford, Diffie e Hellman (1976), propuseram um sistema de criptografia radicalmente novo, no qual as chaves de criptografia e de descriptografia eram diferentes, e a chave de descriptografia não podia ser derivada da chave de criptografia. Em sua proposta, o algoritmo de criptografia (chaveado) E e o algoritmo de descriptografia (chaveado) D tinham de atender a três requisitos, que podem ser declarados da seguinte forma:

1. $D(E(P)) = P$.
2. É extremamente difícil deduzir D a partir de E.
3. E não pode ser decifrado por um ataque de texto simples escolhido.

O primeiro requisito diz que, se aplicarmos D a uma mensagem criptografada, $E(P)$, obteremos outra vez a mensagem de texto simples original P. Sem essa propriedade, o destinatário legítimo não poderia decodificar o texto cifrado. O segundo é autoexplicativo. O terceiro é necessário porque, como veremos em um minuto, os intrusos podem experimentar o algoritmo até se cansarem. Sob essas condições, não há razão para a chave criptográfica não se tornar pública.

O método funciona da seguinte forma: uma pessoa, digamos Alice, desejando receber mensagens secretas, primeiro cria dois algoritmos que atendem aos requisitos anteriores. O algoritmo de criptografia e a chave de Alice se tornam públicos, daí o nome **criptografia de chave pública**. Por exemplo, Alice poderia colocar sua chave pública em sua home page da Web. Usaremos a notação E_A para indicar o algoritmo de criptografia parametrizado pela chave pública de Alice. De modo semelhante, o algoritmo de descriptografia (secreto) parametrizado pela chave privada de Alice é D_A. Bob faz o mesmo, publicando E_B, mas mantendo secreta a chave D_B.

Agora, vamos ver se podemos resolver o problema de estabelecer um canal seguro entre Alice e Bob, que nunca haviam tido um contato anterior. Tanto a chave de criptografia de Alice, E_A, quanto a chave de criptografia de Bob, E_B, estão em arquivos de leitura pública. Agora, Alice pega sua primeira mensagem P, calcula $E_B(P)$ e a envia para Bob. Em seguida, Bob a descriptografa aplicando sua chave secreta D_B [ou seja, ele calcula $D_B(E_B(P)) = P$]. Ninguém mais pode ler a mensagem criptografada $E_B(P)$, porque o sistema de criptografia é considerado sólido e porque é muito difícil derivar D_B da chave E_B publicamente conhecida. Para enviar uma resposta R, Bob transmite $E_A(R)$. Agora, Alice e Bob podem se comunicar com segurança.

Talvez seja interessante fazer uma observação sobre a terminologia. A criptografia de chave pública exige que cada usuário tenha duas chaves: uma chave pública, usada pelo mundo inteiro para criptografar as mensagens a ser enviadas para esse usuário, e uma chave privada, que o usuário utiliza para descriptografar mensagens. Faremos referência a essas chaves como chave *pública* e chave *privada*, respectivamente, e vamos distingui-las das chaves *secretas* (também chamadas chaves simétricas) usadas na criptografia de chave simétrica convencional.

8.3.1 RSA

O único problema é que temos de encontrar algoritmos que realmente satisfaçam todos os três requisitos. Devido às vantagens potenciais da criptografia de chave pública, muitos pesquisadores estão se dedicando integralmente a seu estudo, e alguns algoritmos já foram publicados. Um método muito interessante foi descoberto por um grupo de pesquisadores do MIT (Rivest et al., 1978) e é conhecido pelas iniciais dos três estudiosos que o criaram (Rivest, Shamir, Adleman): **RSA**. Ele sobreviveu a todas as tenta-

tivas de rompimento por mais de um quarto de século e é considerado um algoritmo muito forte. Grande parte da segurança prática se baseia nele. Por isso, Rivest, Shamir e Adleman receberam o ACM Turing Award de 2002. Sua principal desvantagem é exigir chaves de pelo menos 1.024 bits para manter um bom nível de segurança (em comparação com 128 bits para os algoritmos de chave simétrica), e isso o torna bastante lento.

O método RSA se baseia em alguns princípios da teoria dos números. Agora vamos mostrar de forma resumida como usá-lo; para obter mais detalhes, consulte o documento original.

1. Escolha dois números primos grandes, p e q (geralmente, de 1.024 bits).
2. Calcule $n = p \times q$ e $z = (p - 1) \times (q - 1)$.
3. Escolha um número d tal que z e d sejam primos entre si.
4. Encontre e de forma que $e \times d = 1 \bmod z$.

Com esses parâmetros calculados com antecedência, estamos prontos para começar a criptografia. Divida o texto simples (considerado uma sequência de bits) em blocos, de modo que cada mensagem de texto simples P fique no intervalo $0 \leq P < n$. Isso pode ser feito agrupando-se o texto simples em blocos de k bits, onde k é o maior inteiro para o qual a desigualdade $2^k < n$ é verdadeira.

Para criptografar a mensagem P, calcule $C = P^e \pmod n$. Para descriptografar C, calcule $P = C^d \pmod n$. É possível provar que, para todo P na faixa especificada, as funções de criptografia e descriptografia são inversas entre si. Para realizar a criptografia, você precisa de e e n, enquanto para a descriptografia são necessários d e n. Portanto, a chave pública consiste no par (e, n) e a chave privada consiste em (d, n).

A segurança do método se baseia na dificuldade de fatorar números extensos. Se pudesse fatorar o valor n (publicamente conhecido), o criptoanalista poderia então encontrar p e q e, a partir deles, encontrar z. Com o conhecimento de z e e, é possível encontrar d utilizando-se o algoritmo de Euclides. Felizmente, os matemáticos têm tentado fatorar números extensos por pelo menos 300 anos, e o conhecimento acumulado sugere que o problema é extremamente difícil.

De acordo com Rivest e seus colegas, a fatoração de um número de 500 dígitos requer 10^{25} anos, usando-se a força bruta. Nesse caso, eles pressupõem o melhor algoritmo conhecido e um computador com um tempo por instrução de 1 μs. Mesmo que os computadores continuem a se tornar cada vez mais rápidos na proporção de uma ordem de grandeza por década, ainda se passarão séculos até que a fatoração de um número de 500 dígitos se torne viável e, nesse tempo, nossos descendentes poderão simplesmente escolher p e q ainda maiores.

Um exemplo didático muito comum do algoritmo RSA é mostrado na Figura 8.14. Para esse exemplo, escolhemos $p = 3$ e $q = 11$, o que gera $n = 33$ e $z = 20$. Um valor adequado para d é $d = 7$, visto que 7 e 20 não têm fatores comuns. Com essas opções, e pode ser encontrado resolvendo-se a equação $7e = 1 \pmod{20}$, que produz e = 3. O texto cifrado C para uma mensagem de texto simples P é dado por $C = P^3 \pmod{33}$. O texto cifrado é decodificado pelo receptor usando a regra $P = C^7 \pmod{33}$. A figura mostra a codificação do texto simples 'SUZANNE' como exemplo.

Como os números primos escolhidos para esse exemplo são muito pequenos, P tem de ser menor que 33; portanto, cada bloco do texto simples só pode conter um caractere isolado. O resultado é uma cifra de substituição monoalfabética, que não impressiona muito. Se em vez disso tivéssemos escolhido p e $q \approx 2^{512}$, teríamos $n \approx 2^{1024}$ e assim cada bloco poderia ter até 1.024 bits ou 128 caracteres de 8 bits, em comparação com 8 caracteres para o DES e 16 caracteres para o AES.

Devemos destacar que o uso do RSA da forma como descrevemos é semelhante ao uso de um algoritmo simétrico no modo ECB — o mesmo bloco de entrada gera o mesmo bloco de saída. Portanto, é necessário algum tipo de encadeamento para a criptografia de dados. No entanto, na prática, a maior parte dos sistemas baseados no RSA utiliza a criptografia de chave pública, em especial para distribuir chaves únicas de sessão, que, por sua vez, são empregadas com algum algoritmo de chave simétrica, como AES

Texto simples (P)			Texto cifrado (C)		Após a decodificação	
Simbólico	Numérico	P^3	P^3 (mod 33)	C^7	C^7 (mod 33)	Simbólico
S	19	6859	28	13492928512	19	S
U	21	9261	21	1801088541	21	U
Z	26	17576	20	1280000000	26	Z
A	01	1	1	1	01	A
N	14	2744	5	78125	14	N
N	14	2744	5	78125	14	N
E	05	125	26	8031810176	05	E
Cálculo do transmissor				Cálculo do receptor		

Figura 8.14 | Um exemplo do algoritmo RSA.

ou DES triplo. O RSA é lento demais para codificar grandes volumes de dados, mas é amplamente utilizado para a distribuição de chaves.

8.3.2 Outros algoritmos de chave pública

Apesar de ser utilizado em larga escala, o RSA não é de forma alguma o único algoritmo de chave pública conhecido. O primeiro foi o algoritmo da mochila (Merkle e Hellman, 1978). A ideia aqui é que alguém possui um grande número de objetos, cada objeto com um peso diferente. O dono dos objetos codifica a mensagem selecionando secretamente um subconjunto dos objetos e colocando-os na mochila. O peso total dos objetos na mochila torna-se público, e o mesmo acontece com a lista de todos os objetos possíveis. A lista de objetos da mochila é mantida em segredo. Com outras restrições, o problema de descobrir uma lista de objetos possíveis com o peso fornecido foi considerado computacionalmente inviável e formou a base do algoritmo de chave pública.

O inventor do algoritmo, Ralph Merkle, estava bastante seguro de que esse algoritmo não poderia ser decifrado; portanto, ofereceu um prêmio de 100 dólares a quem conseguisse fazê-lo. Adi Shamir (o 'S' do RSA) se prontificou a decifrar o algoritmo e ganhou o prêmio. Indignado, Merkle reforçou o algoritmo e ofereceu um premio de 1.000 dólares a quem pudesse decifrá-lo. Ronald Rivest (o 'R' do RSA) também conseguiu decifrar o novo algoritmo e ganhou o prêmio. Merkle não ousou oferecer 10.000 dólares pela nova versão revisada; portanto, 'A' (Leonard Adleman) não teve sorte. Apesar de ter sido refeito, o algoritmo da mochila não é mais considerado seguro e não é usado na prática.

Outros esquemas de chave pública se baseiam na dificuldade de calcular logaritmos discretos. Os algoritmos que utilizam esse princípio foram criados por El Gamal (1985) e Schnorr (1991).

Existem alguns outros esquemas, como os que se baseiam em curvas elípticas (Menezes e Vanstone, 1993), mas as duas principais categorias são aquelas que se baseiam na dificuldade de fatorar números extensos e no cálculo de logaritmos discretos cuja base é um número primo extenso. Esses problemas são considerados de fato difíceis de resolver — os matemáticos estão estudando os algoritmos há anos sem grandes resultados.

8.4 Assinaturas digitais

A autenticidade de muitos documentos legais, financeiros e outros tipos é determinada pela presença de uma assinatura manual autorizada. Isso não vale para as fotocópias. Para que os sistemas de mensagens computadorizadas possam substituir o transporte físico de documentos em papel e tinta, deve-se encontrar um método que permita assinar os documentos de um modo que não possa ser forjado.

O problema de criar um substituto para as assinaturas escritas à mão é muito difícil. Basicamente, necessita-se de um sistema pelo qual uma parte possa enviar uma mensagem assinada para outra parte de forma que:

1. o receptor possa verificar a identidade alegada pelo transmissor;.
2. mais tarde, o transmissor não possa repudiar o conteúdo da mensagem;
3. o receptor não tenha a possibilidade de inventar ele mesmo a mensagem.

Por exemplo, o primeiro requisito diz respeito a sistemas financeiros. Quando o computador de um cliente pede ao de um banco que compre uma tonelada de ouro, o computador do banco precisa se certificar de que o computador que emitiu o pedido pertence de fato à empresa de cuja conta um valor deve ser debitado. Em outras palavras, os bancos precisam autenticar o cliente (e o cliente precisa autenticar o banco).

O segundo requisito é necessário para proteger o banco contra fraudes. Suponha que o banco compre uma tonelada de ouro e que logo depois disso o preço do ouro caia bruscamente. Um cliente desonesto poderia processar o banco, afirmando nunca ter feito nenhum pedido para a compra de ouro. Quando o banco mostra a mensagem no tribunal, o cliente nega tê-la enviado. A propriedade segundo a qual nenhuma parte de um contrato pode negar mais tarde tê-lo assinado é chamada **não repúdio**. Os esquemas de assinatura digital que estudaremos agora ajudam a garantir o não repúdio.

O terceiro requisito é necessário para proteger o cliente caso o preço do ouro dispare e o banco tente montar uma mensagem assinada na qual o cliente pedia uma barra de ouro em vez de uma tonelada. Nesse cenário de fraude, o banco simplesmente guarda para si próprio o restante do ouro.

8.4.1 Assinaturas de chave simétrica

Uma estratégia para as assinaturas digitais é ter uma autoridade central que saiba de tudo e na qual todos confiem, digamos, Big Brother (*BB*). Em seguida, cada usuário escolhe uma chave secreta e a leva para o escritório do *BB*. Assim, somente Alice e *BB* conhecem a chave secreta de Alice, K_A, e assim por diante.

Quando deseja enviar uma mensagem em texto simples assinada, P, ao gerente de sua conta, que é Bob, Alice gera K_A (B, R_A, t, P), onde B é a identidade de Bob, R_A é um número aleatório escolhido por Alice, t é um registro de tempo para assegurar a atualidade e K_A (B, R_A, t, P) é a mensagem criptografada com sua chave, K_A. Em seguida, ela envia a mensagem, da maneira descrita na Figura 8.15. *BB* vê que essa mensagem veio de Alice, descriptografa a mensagem e a envia a Bob, exatamente como mostramos. A mensagem para Bob contém o texto simples da mensagem de Alice e também a mensagem assinada K_{BB} (A, t, P). Agora Bob executa a solicitação de Alice.

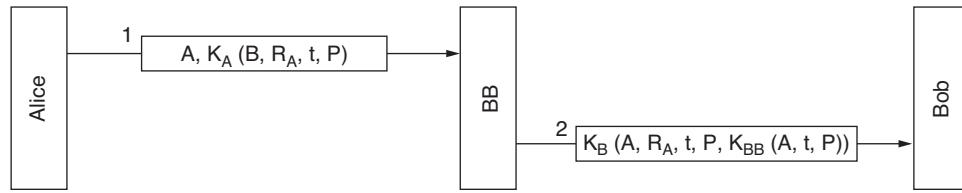



Figura 8.15 | Assinaturas digitais com Big Brother.

O que acontecerá se mais tarde Alice negar ter enviado a mensagem? Na etapa 1, todo mundo processa todo mundo (pelo menos nos Estados Unidos). Por fim, quando o caso parar nos tribunais e Alice continuar negando ter enviado a mensagem a Bob, o juiz perguntará a Bob como ele pode ter certeza de que a mensagem veio de Alice e não de Trudy. Primeiro, Bob destaca que *BB* só aceitará uma mensagem de Alice se ela tiver sido criptografada com K_A. Portanto, não há possibilidade de Trudy ter enviado a *BB* uma mensagem falsa no lugar de Alice, sem que *BB* detectasse esse fato de imediato.

Em seguida, drasticamente, Bob apresenta a Prova A: K_{BB} (A, t, P). Bob afirma tratar-se de uma mensagem assinada por *BB* que prova o fato de Alice ter enviado *P* a Bob. Em seguida, o juiz solicita que *BB* (em quem todos confiam) descriptografe a Prova A. Quando *BB* testemunha que Bob está dizendo a verdade, o juiz decide a favor de Bob. Caso encerrado.

Um problema em potencial com o protocolo de assinatura da Figura 8.15 é Trudy repetir uma das mensagens. Para minimizar esse risco, são utilizados registros de tempo em todas as mensagens. Além disso, Bob pode verificar todas as mensagens mais recentes para ver se R_A foi usada em alguma delas. Caso isso tenha acontecido, a mensagem será descartada por ser uma repetição. Observe que Bob rejeitará as mensagens muito antigas ao verificar seus registros de tempo. Para se proteger contra ataques de repetição repentinos, Bob verifica R_A em cada uma das mensagens recebidas para ver se a mensagem foi enviada por Alice durante a última hora. Caso contrário, Bob pode pressupor com toda segurança que essa solicitação é nova.

8.4.2 Assinaturas de chave pública

Um problema estrutural com o uso da criptografia de chave simétrica para assinaturas digitais é que todos têm de confiar no Big Brother. Além disso, Big Brother tem de ler todas as mensagens assinadas. Os candidatos mais lógicos à execução do servidor Big Brother são o governo, os bancos, os contadores e os advogados. Infelizmente, nenhuma dessas organizações inspira total confiança a todos os cidadãos. Daí, seria interessante se o ato de assinatura de documentos não exigisse a presença de uma autoridade confiável.

Felizmente, a criptografia de chave pública pode trazer uma importante contribuição nessa área. Vamos supor que os algoritmos de criptografia e descriptografia de chave pública tenham a propriedade de que $E(D(P)) = P$ além, é claro, da propriedade habitual de que $D(E(P)) = P$. (O RSA tem essa propriedade; portanto, a suposição não é irracional.) Supondo-se que seja esse o caso, Alice pode enviar uma mensagem de texto simples assinada, *P*, para Bob transmitindo $E_B(D_A\ (P))$. Observe atentamente que Alice conhece sua própria chave de descriptografia (privada), D_A, assim como a chave pública de Bob, E_B; portanto, a criação dessa mensagem é algo que Alice pode fazer.

Quando recebe a mensagem, Bob a transforma usando sua chave privada e produz D_A (P), como mostra a Figura 8.16. Ele guarda esse texto em um lugar seguro e depois aplica E_A para obter o texto simples original.

Para ver como a propriedade de assinatura funciona, suponha que depois Alice negue ter enviado a mensagem *P* para Bob. Quando o caso chegar aos tribunais, Bob po-

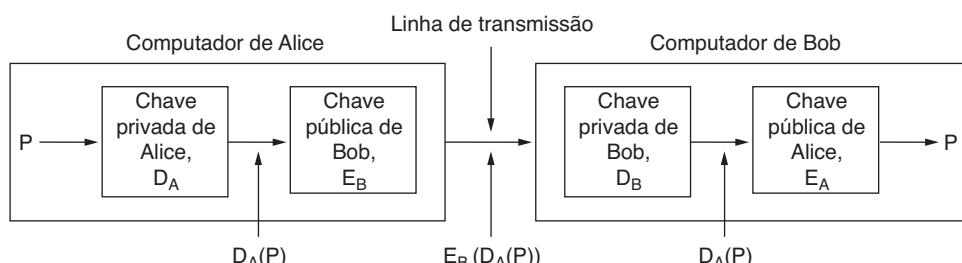



Figura 8.16 | Assinaturas digitais com o uso da criptografia de chave pública.

derá produzir tanto *P* quanto D_A (*P*). O juiz pode confirmar com facilidade que Bob certamente tem uma mensagem válida criptografada por D_A, simplesmente aplicando E_A à mensagem. Como Bob não sabe qual é a chave privada de Alice, a única forma de Bob ter adquirido uma mensagem criptografada por essa chave seria se Alice de fato a tivesse enviado. Enquanto estiver presa por perjúrio e fraude, Alice terá bastante tempo para inventar novos algoritmos de chave pública muito interessantes.

Apesar de a utilização da criptografia de chave pública para assinaturas digitais ser um esquema elegante, há problemas relacionados ao ambiente onde elas operam e não ao algoritmo básico. De um lado, Bob só poderá provar que uma mensagem foi enviada por Alice enquanto D_A permanecer secreta. Se Alice revelar sua chave secreta, o argumento deixará de existir, pois qualquer um poderia ter enviado a mensagem, inclusive o próprio Bob.

Por exemplo, pode ocorrer um problema se Bob for o corretor de ações de Alice. Alice solicita a Bob que ele compre ações ou títulos de determinada empresa. Logo depois disso, o preço cai bruscamente. Para repudiar a mensagem que enviou a Bob, Alice vai à polícia afirmando que sua casa foi assaltada, e o PC que continha sua chave foi roubado. Dependendo das leis do estado ou do país onde mora, ela poderá ou não ser legalmente processada, em especial se afirmar que só descobriu o roubo quando chegou em casa após o trabalho, muitas horas depois do ocorrido.

Outro problema com o esquema de assinatura é o que acontecerá se Alice decidir alterar sua chave. Isso é legal, e provavelmente é uma boa ideia fazê-lo de vez em quando. Se mais tarde surgir um caso jurídico, como descrevemos antes, o juiz aplicará a E_A atual a D_A (*P*) e descobrirá que ela não produz *P*. Nesse momento, a situação de Bob ficará complicada.

Em princípio, qualquer algoritmo de chave pública pode ser usado para assinaturas digitais. O padrão de fato do setor é o algoritmo RSA. Muitos produtos de segurança o utilizam. No entanto, em 1991, o NIST propôs a utilização de uma variante do algoritmo de chave pública de El Gamal em seu novo padrão, o **DSS** (**Digital Signature Standard**). O El Gamal obtém sua segurança a partir da dificuldade de calcular logaritmos discretos, e não da dificuldade de fatorar números muito extensos.

Como sempre acontece quando o governo tenta ditar padrões criptográficos, houve um tumulto. O DSS foi criticado por ser:

1. secreto demais (a NSA projetou o protocolo para utilizar o El Gamal).
2. lento demais (10 a 40 vezes mais lento do que o RSA para verificar assinaturas).
3. novo demais (o El Gamal ainda não foi analisado por completo).
4. inseguro demais (chave fixa de 512 bits).

Em uma versão posterior, o quarto ponto rendeu muita discussão quando foram permitidas chaves com até 1.024 bits. Apesar disso, os dois primeiros pontos permanecem válidos.

8.4.3 Sumário de mensagens

Uma crítica aos métodos de assinatura é que eles frequentemente reúnem duas funções distintas: autenticação e sigilo. Em geral, a autenticação é necessária, mas o sigilo não. Além disso, obter uma licença para exportação normalmente é mais fácil se o sistema em questão oferecer apenas autenticação, mas não sigilo. A seguir, descreveremos um esquema de autenticação que não exige a criptografia da mensagem inteira.

Esse esquema se baseia na ideia de uma função de hash unidirecional que extrai um trecho qualquer do texto simples e, a partir dele, calcula uma sequência de bits de tamanho fixo. Essa função de hash, representada por *MD* (Message Digest), geralmente é chamada **sumário da mensagem** e tem quatro propriedades importantes:

1. se *P* for fornecido, o cálculo de *MD*(*P*) será muito fácil.
2. se *MD*(*P*) for fornecido, será efetivamente impossível encontrar *P*.
3. dado *P*, ninguém pode encontrar *P'* tal que *MD*(*P'*) = *MD*(*P*).
4. uma mudança na entrada de até mesmo 1 bit produz uma saída muito diferente.

Para atender ao critério 3, a função de hash deve ter pelo menos 128 bits, de preferência mais. Para atender ao critério 4, o hash deve desfigurar completamente os bits, o que não é diferente dos algoritmos de criptografia de chave simétrica que vimos.

Calcular um sumário da mensagem a partir de um trecho de texto simples é muito mais rápido que criptografar esse texto simples com um algoritmo de chave pública; portanto, os sumários de mensagens podem ser usados para agilizar algoritmos de assinatura digital. Para ver como isso funciona, considere mais uma vez o protocolo de assinatura da Figura 8.15. Em vez de assinar *P* com K_{BB} (*A*, *t*, *P*), agora *BB* calcula o sumário da mensagem aplicando *MD* a *P*, produzindo *MD*(*P*). Em seguida, *BB* inclui K_{BB} (*A*, *t*, *MD*(*P*)) como o quinto item da lista criptografada com K_B que é enviada a Bob, em vez de K_{BB} (*A*, *t*, *P*).

Se houver uma contestação, Bob poderá produzir tanto *P* quanto K_{BB} (*A*, *t*, *MD*(*P*)). Depois que Big Brother tiver descriptografado o item para o juiz, Bob terá *MD*(*P*) que, certamente, é genuíno, além do *P* alegado. No entanto, como é impossível para Bob encontrar outra mensagem que produza esse hash, o juiz será facilmente convencido de que Bob está falando a verdade. A utilização de sumários de mensagens dessa forma poupa tempo de criptografia e reduz os custos com o transporte de mensagens.

Sumários de mensagens também funcionam em sistemas de criptografia de chave pública, como mostra a Figura 8.17. Aqui, primeiro Alice calcula o sumário da mensagem de seu texto simples. Em seguida, ela assina a mensagem e envia tanto a compilação assinada quando o texto simples a Bob. Se Trudy substituir *P* durante a operação, Bob perceberá a troca quando calcular *MD*(*P*).

SHA-1 e SHA-2

Diversas funções para o sumário da mensagem foram propostas. Uma das funções mais utilizadas é **SHA-1** (**Secure Hash Algorithm 1**) (NIST, 1993). Como em todos os sumários de mensagens, ele opera tratando os bits de uma maneira suficientemente complicada, de modo que cada bit da saída seja afetado por cada bit da entrada. SHA-1 foi desenvolvimento pela NSA e ratificado pelo NIST no FIPS 180-1. Ele processa os dados de entrada em blocos de 512 bits e gera um sumário da mensagem de 160 bits. Um modo típico de Alice enviar uma mensagem não secreta mas assinada para Bob é ilustrado na Figura 8.18. Nessa figura, sua mensagem de texto simples é colocada no algoritmo SHA-1 para obter um hash de 160 bits do SHA-1. Em seguida, Alice assina o hash com sua chave privada RSA e envia a mensagem de texto simples e o hash assinado para Bob.

Figura 8.17 | Assinaturas digitais utilizando sumários de mensagens.

Depois de receber a mensagem, Bob calcula o hash SHA-1 e também aplica a chave pública de Alice ao hash assinado para obter o original, *H*. Se os dois corresponderem, a mensagem será considerada válida. Tendo em vista que não há nenhum meio para Trudy modificar a mensagem (de texto simples) enquanto ela estiver em trânsito e produzir uma nova mensagem com hash *H*, Bob pode detectar com facilidade quaisquer mudanças que Trudy tenha feito na mensagem. Para mensagens cuja integridade seja importante, mas cujo conteúdo não seja secreto, o esquema da Figura 8.18 é bastante utilizado. Por um custo relativamente pequeno em computação, ele garante que as modificações feitas na mensagem de texto simples em trânsito poderão ser detectadas com probabilidade muito alta.

Agora, vamos ver rapidamente como o SHA-1 funciona. Ele começa preenchendo a mensagem com a adição de um bit 1 ao final, seguido pelo número de bits 0 necessários para tornar o tamanho um múltiplo de 512 bits. Em seguida, um número de 64 bits contendo o tamanho da mensagem antes do preenchimento é submetido a uma operação OR nos 64 bits de baixa ordem. Na Figura 8.19, a mensagem é mostrada com o preenchimento à direita, porque o texto em inglês e as figuras se estendem da esquerda para a

Figura 8.18 | Utilização do SHA-1 e do RSA para assinar mensagens não secretas.

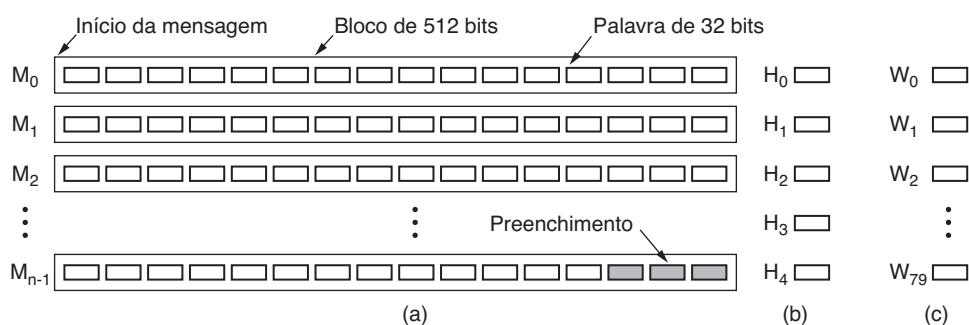



Figura 8.19 | (a) Uma mensagem preenchida até um multiplo de 512 bits. (b) As variáveis de saída. (c) O array de palavras.

direita (isto é, o canto inferior direito em geral é percebido como o fim da figura). Com os computadores, essa orientação corresponde a máquinas big-endian, como SPARC e IBM 360, e seus sucessores, mas o SHA-1 sempre preenche o fim da mensagem, não importando que máquina endian seja usado.

Durante o cálculo, o SHA-1 mantém cinco variáveis de 32 bits, de H_0 até H_4, onde o hash se acumula. Estas são mostradas na Figura 8.19(b). Elas são inicializadas como constantes especificadas no padrão.

Cada um dos blocos de M_0 até M_{n-1} agora é processado um por vez. Para o bloco atual, as 16 palavras são primeiro copiadas para o início de um array auxiliar de 80 palavras, W, como mostra a Figura 8.19(c). Depois, as outras 64 palavras em W são preenchidas usando a fórmula

$$W_i = S^1\ (W_{i-3} \text{ XOR } W_{i-8} \text{ XOR } W_{i-14} \text{ XOR } W_{i-16}) \quad (16 \leq i \leq 79)$$

onde $S^b(W)$ representa a rotação circular à esquerda da palavra de 32 bits, W, por b bits. Agora, cinco variáveis auxiliares, de A até E, são iniciadas de H_0 até H_4, respectivamente.

O cálculo real pode ser expresso em pseudo-C como:

```
for (i = 0; i < 80; i++) {
    temp = S5(A) + fi (B, C, D) + E + Wi + Ki;
    E = D;  D = C;  C = S30(B);  B = A;  A = temp;
}
```

onde as constantes K_i são definidas no padrão. As funções f_i são definidas como:

$$f_i\,(B,C,D) = (B \text{ AND } C) \text{ OR } (\text{NOT } B \text{ AND } D) \quad (0 \leq i \leq 19)$$

$$f_i\,(B,C,D) = B \text{ XOR } C \text{ XOR } D \quad (20 \leq i \leq 39)$$

$$f_i\,(B,C,D) = (B \text{ AND } C) \text{ OR } (B \text{ AND } D) \text{ OR } (C \text{ AND } D) \quad (40 \leq i \leq 59)$$

$$f_i\,(B,C,D) = B \text{ XOR } C \text{ XOR } D \quad (60 \leq i \leq 79)$$

Quando todas as 80 iterações do loop são completadas, as variáveis de A até E são somadas a H_0 até H_4, respectivamente.

Agora que o primeiro bloco de 512 bits foi processado, o próximo é iniciado. O array W é reiniciado a partir do novo bloco, mas H fica como estava. Quando esse bloco é concluído, o próximo é iniciado, e assim por diante, até todos os blocos da mensagem de 512 bits serem processados. Quando o último bloco é concluído, as cinco palavras de 32 bits no array H são transmitidas como saída, formando o hash criptográfico de 160 bits. O código C completo para SHA-1 é dado na RFC 3174.

Novas versões de SHA-1 foram desenvolvidas para produzir hashes de 224, 256, 384 e 512 bits. Coletivamente, essas versões são chamadas SHA-2. Não apenas esses são hashes maiores do que hashes SHA-1, mas a função de sumário foi mudada para combater alguns pontos fracos em potencial do SHA-1. O SHA-2 ainda não é muito usado, mas provavelmente o será no futuro.

MD5

Para completar, vamos mencionar outro sumário que é muito popular. **MD5** (Rivest, 1992) é o quinto de uma série de sumários de mensagens projetados por Ronald Rivest. Resumindo, a mensagem é preenchida para um tamanho de 448 bits (módulo 512). Depois, o comprimento original da mensagem é acrescentado como um inteiro de 64 bits para gerar uma entrada total cujo comprimento é um múltiplo de 512 bits. Em cada rodada, um bloco de entrada de 512 bits é extraído e colocado no buffer de 128 bits. Para que os cálculos sejam feitos com maior precisão, também é incluída uma tabela criada a partir da função seno. O objetivo da utilização de uma função conhecida é evitar qualquer suspeita de que o projetista tenha criado uma armadilha secreta para seu próprio uso. Esse processo continua até que todos os blocos de entrada tenham sido consumidos. O conteúdo do buffer de 128 bits forma o sumário de mensagens.

Após mais de uma década de uso sólido e estudo, os pontos fracos no MD5 têm levado à capacidade de encontrar colisões, ou então diferentes mensagens com o mesmo hash (Sotirov et al., 2008). Esse é o prenúncio do fim de uma função de sumário, pois significa que o sumário não pode ser usado com segurança para representar uma mensagem. Assim, a comunidade de segurança classifica o MD5 como quebrado; ele deve ser substituído sempre que possível, e nenhum sistema novo deverá usá-lo como parte de seu projeto. Apesar disso, você ainda poderá encontrar o MD5 nos sistemas existentes.

8.4.4 O ataque do aniversário

No mundo da criptografia, nada é o que parece. Talvez você esteja pensando que são necessárias aproximadamente 2^m operações para subverter um sumário da mensagem de m bits. Na verdade, normalmente $2^{m/2}$ operações serão suficientes, utilizando-se o **ataque do aniversário**, um método publicado por Yuval (1979) no clássico artigo 'How to Swindle Rabin' (Como enganar Rabin).

A ideia para esse ataque vem de uma técnica que frequentemente os professores de matemática utilizam em seus cursos de probabilidade. A pergunta é a seguinte: quantos alunos você deverá ter em uma sala de aula para que a probabilidade de haver duas pessoas que aniversariem no mesmo dia exceda 1/2? A maioria dos alunos espera que a resposta seja superior a 100. Na verdade, a teoria da probabilidade afirma que esse número é apenas 23. Intuitivamente, sem fazer uma análise rigorosa, com 23 pessoas podemos formar (23 × 22)/2 = 253 pares diferentes, cada um dos quais tem uma probabilidade igual a 1/365 de ser um acerto. Sob esse aspecto, essa probabilidade deixa de ser tão surpreendente.

De modo mais geral, se houver algum mapeamento entre as entradas e as saídas, com n entradas (pessoas, mensagens etc.) e k saídas possíveis (aniversários, sumários de mensagens etc.), haverá $n(n - 1)/2$ pares de entrada. Se $n(n - 1)/2 > k$, a chance de haver pelo menos uma correspondência será muito boa. Portanto, fazendo uma aproximação, é provável que haja uma correspondência para $n > \sqrt{k}$. Esse resultado significa que provavelmente um sumário da mensagem de 64 bits possa ser quebrado gerando-se 2^{32} mensagens e procurando-se duas mensagens com o mesmo sumário.

Agora, vejamos um exemplo prático. O departamento de ciência da computação da State University nos Estados Unidos tem uma única vaga estável para dois professores, Tom e Dick. Tom foi admitido dois anos antes de Dick e, portanto, é convocado para os testes primeiro. Se ele for aprovado, Dick será descartado. Tom sabe que a chefe de seu departamento, Marilyn, gosta muito do trabalho dele; sendo assim, ele pede a Marilyn que escreva uma carta de recomendação ao reitor, que tomará a decisão a respeito do cargo. Depois de enviadas, todas as cartas passam a ser confidenciais.

Marilyn pede a sua secretária, Ellen, que escreva a carta para o reitor, fazendo um esboço do que deseja. Quando a carta está pronta, Marilyn a revisa, calcula e assina o sumário de 64 bits e o envia ao reitor. Ellen pode enviar a carta mais tarde pelo correio eletrônico.

Infelizmente para Tom, Ellen está envolvida emocionalmente com Dick e gostaria que Tom fosse descartado; assim, escreve a carta a seguir com as 32 opções entre colchetes.

Caro Sr. Reitor,

Esta [*carta* | *mensagem*] tem como objetivo expressar minha [*honesta* | *franca*] opinião a respeito do professor Tom Wilson, que [*é* | *está*] [*candidato* | *prestes*] [*para* | *a*] obter uma vaga permanente nesta universidade [*imediatamente* | *este ano*]. Eu [*conheço* | *trabalho com*] o professor Wilson há seis anos. Ele é um [*destacado* | *excelente*] pesquisador de grande [*talento* | *capacidade*] [*mundialmente* | *internacionalmente*] conhecido por suas [*brilhantes* | *criativas*] ideias a respeito de [*muitos* | *uma grande variedade de*] problemas [*difíceis* | *complicados*].

Ele também é um [*professor* | *educador*] [*bastante* | *muito*] [*respeitado* | *admirado*]. Seus alunos fazem críticas [*maravilhosas* | *espetaculares*] de [*suas aulas* | *seus cursos*]. Ele é o [*professor* | *orientador*] [*mais querido* | *mais conhecido*] [*da universidade* | *do departamento*].

[*Além disso* | *Além do mais*], o professor Wilson é um [*grande* | *fantástico*] administrador. [*Seus contratos* | *Suas concessões*] trouxeram uma [*grande* | *substancial*] quantia em dinheiro para [*o* | *nosso*] departamento. [*Esse dinheiro* | *Esses fundos*] [*permitiu* | *permitiram*] que [*criássemos* | *realizássemos*] muitos programas [*especiais* | *importantes*], [*tais como* | *por exemplo*] o programa Universidade 2000. Sem esses fundos, [*seríamos incapazes* | *não seríamos capazes*] de dar continuidade a esse programa, que é tão [*importante* | *essencial*] para nós. Afirmo ao senhor que ele é o profissional mais adequado para essa posição.

Infelizmente para Tom, assim que acaba de redigir e digitar essa carta, Ellen também digita a seguinte carta:

Caro Sr. Reitor,

Esta [*carta* | *mensagem*] tem como objetivo expressar minha [*honesta* | *franca*] opinião a respeito do professor Tom Wilson, que [*é* | *está*] [*candidato* | *prestes*] [*para* | *a*] assumir uma vaga permanente nesta universidade [*imediatamente* | *este ano*]. Eu [*conheço* | *trabalho com*] Tom há seis anos. Ele é um [*incompetente* | *mau*] pesquisador, não é bem visto em sua [*especialidade* | *área*]. Sua pesquisa [*raramente* | *esporadicamente*] mostra [*bom-senso* | *conhecimento*] dos [*principais* | *mais importantes*] problemas atuais.

Ele não é um [*professor* | *educador*] [*bastante* | *muito*] [*respeitado* | *admirado*]. Seus alunos fazem [*duras* | *pesadas*] críticas de [*suas aulas* | *seus cursos*]. Ele é o [*professor* | *orientador*] mais impopular [*da universidade* | *do departamento*] devido à sua [*tendência* | *propensão*] a [*ridicularizar* | *embaraçar*] os alunos que fazem perguntas em suas aulas.

[*Além disso* | *Além do mais*], Tom é um administrador [*terrível* | *fraco*]. [*Seus contratos* | *Suas concessões*] trouxeram apenas uma [*insignificante* | *pequena*] quantia em dinheiro para [*o* | *nosso*] departamento. A menos que mais [*verbas* | *fundos*] sejam [*alocadas* | *alocados*], teremos de cancelar alguns programas essenciais, tais como seu programa Universidade 2000. Infelizmente, sob essas [*condições* | *circunstâncias*], não posso recomendá-lo em sã consciência para essa posição.

Ellen passa o programa em seu computador para calcular os 2^{32} sumários de mensagens de cada carta. Há chances de que um sumário da primeira carta corresponda a um sumário da segunda carta. Caso isso não aconteça, ela poderá incluir algumas outras opções e tentar de novo durante o fim de semana. Suponha que ela encontre uma correspondência. Vamos chamar a carta 'boa' de *A* e a 'ruim' de *B*.

Em seguida, pelo correio eletrônico, Ellen envia a carta *A* a Marilyn para que seja aprovada. Ela mantém a carta B completamente secreta, sem mostrá-la a ninguém. É claro que Marilyn aprova a carta, calcula seu sumário da mensagem de 64 bits, assina o sumário e envia-o assinado ao reitor Smith utilizando o correio eletrônico. Por outro lado, Ellen envia a carta *B* ao reitor (não a carta *A*, como deveria fazer).

Depois de obter a carta e o sumário da mensagem assinado, o reitor executa o algoritmo de sumário da mensagem na carta *B*, observa que ela corresponde ao sumário que Marilyn enviou e despede Tom. O reitor não percebe que Ellen gerou duas cartas com o mesmo sumário da mensagem e enviou a ele uma mensagem diferente da que Marylin viu e aprovou. (Final opcional: Ellen conta a Dick o que fez. Dick não gosta do que ouve e termina o namoro com ela. Ellen fica furiosa e confessa tudo a Marilyn. Marilyn telefona para o reitor. Tom acaba ficando com o cargo.) Com o SHA-1, o ataque do aniversário se torna impraticável, pois, mesmo com um trilhão de sumários por segundo, seriam necessários 32 mil anos para calcular todos os 2^{80}

sumários de duas cartas com 80 variantes cada uma e, de qualquer forma, não poderíamos ter certeza de que haveria uma correspondência. É claro que, com uma nuvem de 1 milhão de chips trabalhando em paralelo, 32 mil anos se transformam em duas semanas.

8.5 Gerenciamento de chaves públicas

A criptografia de chave pública torna possível a comunicação segura a pessoas que não compartilham uma chave comum, e também possibilita a assinatura de mensagens sem a presença de uma terceira parte confiável. Finalmente, os sumários assinados das mensagens permitem verificar com facilidade e segurança a integridade de mensagens recebidas.

Porém, existe um problema que ignoramos até aqui: se Alice e Bob não conhecem um ao outro, como eles vão obter as respectivas chaves públicas para iniciar o processo de comunicação? A solução óbvia — colocar a chave pública no site — não funciona pela seguinte razão: suponha que Alice queira pesquisar a chave pública de Bob em seu site. Como ela fará isso? Bem, Alice começa digitando o URL de Bob. Seu navegador então pesquisa o endereço DNS da home page de Bob e lhe envia uma solicitação *GET*, como mostra a Figura 8.20. Infelizmente, Trudy intercepta a solicitação e responde com uma home page falsa, talvez uma cópia da de Bob, exceto pela substituição da chave pública de Bob pela chave pública de Trudy. Quando Alice codifica sua primeira mensagem com E_T, ela a decodificará, lerá e recodificará com a chave pública de Bob, enviando a mensagem a Bob, que não sabe que ela está lendo suas mensagens recebidas. Pior ainda, Trudy poderia modificar as mensagens antes de recodificá-las para Bob. É claro que há necessidade de algum mecanismo para garantir que as chaves públicas possam ser trocadas em segurança.

8.5.1 Certificados

Como uma primeira tentativa de distribuição de chaves públicas com segurança, poderíamos imaginar um centro de distribuição de chaves, **KDC** (**Key Distribution Center**), disponível on-line 24 horas por dia, a fim de fornecer chaves públicas por demanda. Um dos muitos problemas com essa solução é o fato de ela não ser escalável, e o centro de distribuição de chaves rapidamente se tornaria um gargalo. Além disso, se ele ficasse inativo, a segurança da Internet seria paralisada repentinamente.

Por essas razões, as pessoas desenvolveram uma solução diferente, que não exige que o centro de distribuição de chaves esteja on-line todo o tempo. De fato, ele não precisa estar on-line de modo algum. Em vez disso, ele certifica as chaves públicas pertencentes a pessoas, empresas e outras organizações. Uma organização que certifica chaves públicas é chamada autoridade de certificação, ou **CA** (**Certification Authority**).

Como um exemplo, suponha que Bob queira permitir que Alice e outras pessoas se comuniquem com ele com segurança. Ele pode ir à CA com sua chave pública e seu passaporte ou algum outro documento de identidade e solicitar a certificação. A CA emite então um certificado semelhante ao da Figura 8.21 e assina seu hash SHA-1 com a chave privada da CA. Em seguida, Bob paga a taxa da CA e obtém um CD contendo o certificado e seu hash assinado.

A principal função de um certificado é vincular uma chave pública ao nome de um protagonista (indivíduo, empresa etc.). Os certificados em si não são secretos ou protegidos. Por exemplo, Bob poderia decidir colocar seu novo certificado em seu site, com um link na página principal informando: clique aqui para ver meu certificado de chave pública. O clique resultante retornaria o certificado e o bloco de assinatura (o hash SHA-1 assinado do certificado).

Agora, vamos percorrer o cenário da Figura 8.20 novamente. Quando Trudy intercepta a solicitação de Alice para a home page de Bob, o que ela pode fazer? Trudy pode inserir seu próprio certificado e seu bloco de assinatura na página falsa; porém, quando Alice ler o certificado, verá imediatamente que não está se comunicando com Bob, porque o nome de Bob não está no certificado. Trudy pode modificar a home page de Bob durante a execução, substituindo a chave pública de Bob por sua própria chave. Porém, quando Alice executar o algoritmo SHA-1 no certificado, ela obterá um hash que não corresponde ao que ela recebe ao aplicar a chave pública conhecida da CA ao bloco de assinatura. Como Trudy não tem a chave privada da CA, ela não tem meios de gerar um bloco de assinatura que contenha o hash da página Web modificada com sua chave pública. Desse modo, Alice pode estar certa de que possui a chave pública de Bob e não a de Trudy ou de outra pessoa. Como afirmamos, esse esquema não exige que a CA esteja on-line para verificação, eliminando assim um gargalo em potencial.

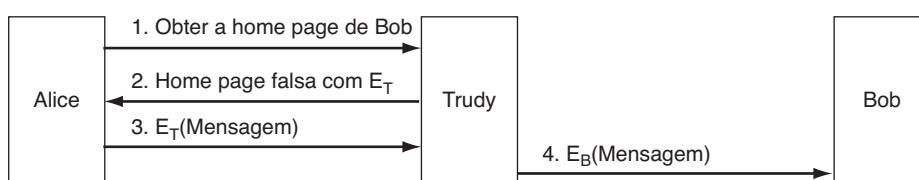



Figura 8.20 | Um modo de Trudy subverter a criptografia de chave pública.

Certifico que a chave pública
19836A8B03030CF83737E3837837FC3s7092827262643FFA82710382828282A
pertence a
João Roberto da Silva
Avenida Brasil, 12345
Rio das Ostras, RJ 28890-000
Nascimento: 7 de setembro de 1958
E-mail: bob@supernet.com.br

Hash SHA-1 do certificado acima assinado com a chave privada da CA

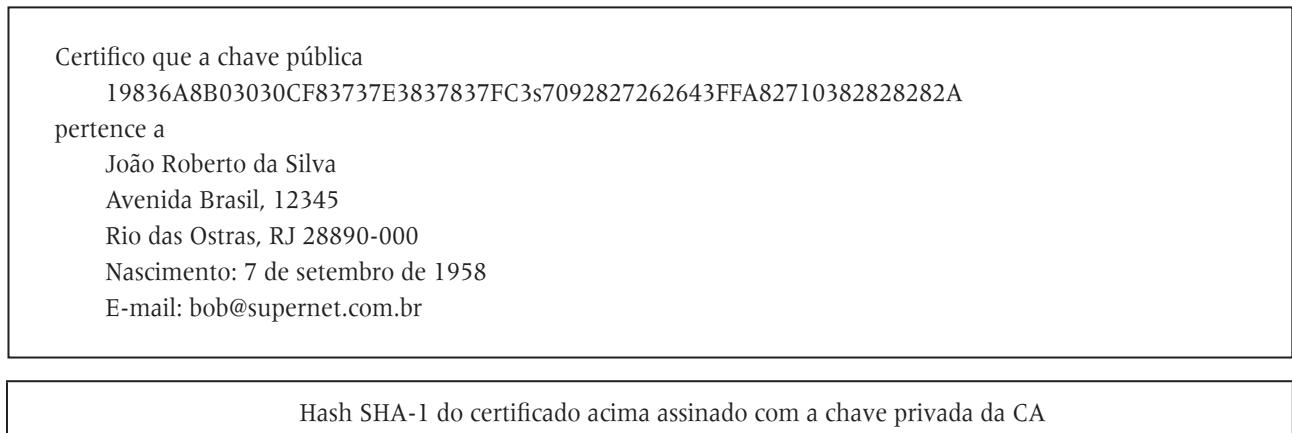



Figura 8.21 | Um certificado possível e seu hash assinado.

Embora a função-padrão de um certificado seja vincular uma chave pública a um protagonista, um certificado também pode ser usado para vincular uma chave pública a um atributo. Por exemplo, um certificado poderia afirmar: 'esta chave pública pertence a alguém com mais de 18 anos'. Ela pode ser usada para provar que o proprietário da chave privada não é menor de idade e, portanto, pode acessar material não apropriado para crianças, e assim por diante, mas sem revelar a identidade do proprietário. Em geral, a pessoa que tivesse o certificado o enviaria ao site, ao protagonista ou ao processo que solicitasse informações sobre a idade. Esse site, protagonista ou processo geraria então um número aleatório e o codificaria com a chave pública no certificado. Se o proprietário fosse capaz de decodificá-lo e enviá-lo de volta, essa seria a prova de que o proprietário de fato tinha o atributo declarado no certificado. Como alternativa, o número aleatório poderia ser usado para gerar uma chave de sessão pela duração da conversação.

Outro exemplo de situação em que um certificado poderia conter um atributo é um sistema distribuído orientado a objetos. Em geral, cada objeto tem diversos métodos. O proprietário do objeto poderia fornecer a cada cliente um certificado com um mapa de bits dos métodos que o cliente tem permissão para invocar e vincular o mapa de bits a uma chave pública, usando um certificado assinado. Mais uma vez, se o proprietário do certificado puder provar a posse da chave privada correspondente, ele terá permissão para executar os métodos no mapa de bits. A identidade do proprietário não precisa ser conhecida, uma característica útil em situações nas quais a privacidade é importante.

8.5.2 X.509

Se todos os que quisessem algo assinado fossem à CA com um tipo de certificado diferente, logo se tornaria um problema administrar todos os formatos diferentes. Para resolver esse problema, foi criado e aprovado pela ITU um padrão para certificados. O padrão é chamado **X.509** e seu uso está difundido na Internet. Ele passou por três versões desde a padronização inicial, em 1988. Vamos descrever a V3.

O X.509 foi muito influenciado pelo mundo OSI, e toma emprestadas algumas de suas piores características (por exemplo, nomenclatura e codificação). De modo surpreendente, a IETF aceitou o X.509, embora em quase todas as outras áreas — desde endereços de máquina até protocolos de transporte e formatos de correio eletrônico — ela tenha ignorado o OSI e tentado fazer tudo da maneira certa. A versão da IETF do X.509 é descrita na RFC 3280.

Em seu núcleo, o X.509 é um modo de descrever certificados. Os principais campos em um certificado estão listados na Tabela 8.3. As descrições dadas na tabela devem fornecer uma ideia geral do significado dos campos. Para obter mais informações, consulte o próprio padrão ou a RFC 2459.

Por exemplo, se Bob trabalhasse no departamento de empréstimos do Money Bank, seu endereço X.500 poderia ser:

/C=US/O=MoneyBank/OU=Loan/CN=Bob/

onde *C* é o país, *O* é a organização, *OU* é a unidade organizacional e *CN* é o nome comum. As CAs e outras entidades são identificadas de modo semelhante. Um problema significativo com os nomes X.500 é que, se Alice estiver tentando entrar em contato com bob@moneybank.com e receber um certificado com um nome X.500, talvez não fique claro para ela a que Bob o certificado se refere. Felizmente, a partir da versão 3, os nomes DNS são permitidos, em lugar de nomes X.500; assim, esse problema por fim deve desaparecer.

Os certificados são codificados com o uso da **ASN.1 (Abstract Syntax Notation 1)** da OSI, que pode ser considerada uma struct em C, exceto por sua notação muito peculiar e extensa. Para obter mais informações sobre o X.509, consulte Ford e Baum (2000).

Campo	Significado
Version	A versão do X.509
Serial number	Este número, somado ao nome da CA, identifica o certificado de forma exclusiva
Signature algorithm	O algoritmo usado para assinar o certificado
Issuer	Nome X.500 da CA
Validity period	Períodos inicial e final de validade
Subject name	A entidade cuja chave está sendo certificada
Public key	A chave pública da entidade certificada e a ID do algoritmo utilizado
Issuer ID	Uma ID opcional que identifica de forma exclusiva o emissor do certificado
Subject ID	Uma ID opcional que identifica de forma exclusiva a entidade certificada
Extensions	Muitas extensões foram definidas
Signature	A assinatura do certificado (assinado pela chave privada da CA)

Tabela 8.3 | Os campos básicos de um certificado X.509.

8.5.3 Infraestruturas de chave pública

Fazer uma única CA emitir todos os certificados do mundo é algo que com certeza não funcionaria. Ela entraria em colapso sob a carga e também seria um ponto central de falha. Uma solução possível poderia ser a existência de várias CAs, todas administradas pela mesma organização e todas usando a mesma chave privada para assinar certificados. Embora isso pudesse resolver o problema da carga e da falha, há um novo problema: o vazamento de chaves. Se houvesse dezenas de servidores espalhados pelo mundo, todos com a chave privada da CA, o risco de que a chave privada fosse roubada ou sofresse algum outro tipo de vazamento seria bastante aumentado. Tendo em vista que o comprometimento dessa chave arruinaria a infraestrutura de segurança eletrônica do mundo, a existência de uma única CA central é muito arriscada.

Além disso, que organização operaria a CA? É difícil imaginar uma autoridade que fosse aceita em todo o mundo como uma entidade legítima e confiável. Em alguns países, as pessoas insistiriam em que essa entidade fosse o governo; em outros países, elas insistiriam em que não fosse o governo.

Por essas razões, foi desenvolvido um modo diferente de certificar chaves públicas, identificado pelo nome geral **PKI** (**Public Key Infrastructure**). Nesta seção, resumiremos como ele funciona em linhas gerais, embora existam muitas propostas relativas aos detalhes que provavelmente vão evoluir com o tempo.

Uma PKI tem vários componentes, incluindo usuários, CAs, certificados e diretórios. A função da PKI é fornecer um modo de estruturar esses componentes e definir padrões para os vários documentos e protocolos. Uma forma particularmente simples de PKI é uma hierarquia de CAs, como mostra a Figura 8.22. Nesse exemplo, apresentamos três níveis, mas, na prática, pode haver um número menor ou maior. A CA de nível superior, chamada raiz, certifica CAs do segundo nível, que denominamos **RAs (Regional Authorities)**, porque podem cobrir alguma região geográfica, como um país ou um continente. Entretanto, esse termo não é padrão; de fato, nenhum termo é realmente padrão para os diferentes níveis da árvore. Por sua vez, as RAs certificam as CAs reais, que emitem os certificados X.509 para organizações e indivíduos. Quando a raiz autoriza uma nova RA, ela gera um certificado X.509 anunciando que aprovou a RA, inclui a chave pública da nova RA no certificado, assina o certificado e o entrega à RA. De modo semelhante, quando uma RA aprova uma nova CA, ela produz e assina um certificado declarando sua aprovação e contendo a chave pública da CA.

Nossa PKI funciona de modo semelhante. Suponha que Alice precise da chave pública de Bob, a fim de se comunicar com ele; então, ela procura e encontra um certificado contendo a chave, assinado pela CA 5. Porém, Alice nunca ouviu falar da CA 5. Pelo que ela sabe, a CA 5 poderia ser a filha de 10 anos de Bob. Ela poderia ir até a CA 5 e dizer: prove sua legitimidade. A CA 5 responde com o certificado que recebeu da RA 2, que contém a chave pública da CA 5. Agora, munida da chave pública da CA 5, Alice pode confirmar que o certificado de Bob foi de fato assinado pela CA 5 e, portanto, é válido.

A menos que a RA 2 seja o filho de 12 anos de Bob. Nesse caso, a próxima etapa é pedir à RA 2 que prove sua legitimidade. A resposta à consulta de Alice é um certificado assinado pela raiz contendo a chave pública da RA 2. Agora, Alice tem certeza de que possui a chave pública de Bob.

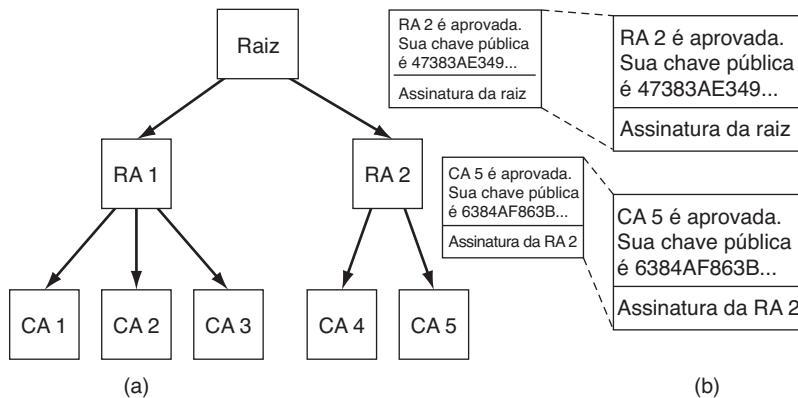



Figura 8.22 | (a) Uma PKI hierárquica. (b) Uma cadeia de certificados.

No entanto, como Alice encontra a chave pública da raiz? Por mágica. Supõe-se que todo mundo conheça a chave pública da raiz. Por exemplo, seu navegador pode ter sido distribuído já contendo a chave pública da raiz.

Bob é o tipo de sujeito amigável e não quer dar muito trabalho a Alice. Ele sabe que Alice vai ter de verificar a CA 5 e a RA 2; assim, para evitar dificuldades, ele reúne os dois certificados necessários e os fornece a ela junto com seu próprio certificado. Agora, ela pode usar seu conhecimento da chave pública da raiz para confirmar o certificado de nível superior e a chave pública que ele contém para verificar o segundo certificado. Desse modo, Alice não precisa entrar em contato com ninguém para fazer a verificação. Como os certificados são todos assinados, ela pode detectar com facilidade quaisquer tentativas de falsificar seu conteúdo. Uma cadeia de certificados como essa, que volta à raiz, às vezes é chamada **corrente de confiança** ou **caminho de certificação**. A técnica é amplamente utilizada na prática.

É claro que ainda temos o problema de saber quem vai administrar a raiz. A solução é não haver uma única, mas sim várias raízes, cada uma com suas próprias RAs e CAs. De fato, os navegadores modernos são previamente carregados com as chaves públicas de mais de 100 raízes, às vezes referidas como **âncoras de confiança**. Desse modo, pode-se evitar ter uma única autoridade confiável no mundo inteiro.

Entretanto, agora existe a questão de como o fornecedor do navegador decide quais das supostas âncoras de confiança são de fato confiáveis e quais são desprezíveis. Tudo se reduz à confiança do usuário no fornecedor do navegador para fazer escolhas sensatas e não aprovar simplesmente todas as âncoras de confiança dispostas a pagar por sua inclusão na lista. A maioria dos navegadores permite que os usuários inspecionem as chaves da raiz (em geral, sob a forma de certificados assinados pela raiz) e eliminem qualquer uma que parecer obscura.

Diretórios

Outra questão importante para qualquer PKI é onde estão armazenados os certificados (e suas cadeias de retorno até alguma âncora de confiança conhecida). Uma possibilidade é fazer cada usuário armazenar seus próprios certificados. Embora isso seja seguro (isto é, não existe nenhum meio de os usuários adulterarem certificados assinados sem detecção), também é inconveniente. Uma alternativa proposta é usar o DNS como um diretório de certificados. Antes de entrar em contato com Bob, é provável que Alice tenha de pesquisar seu endereço IP usando o DNS; então, por que não fazer o DNS retornar toda a cadeia de certificados de Bob juntamente com seu endereço IP?

Algumas pessoas acham que essa é a melhor alternativa, mas outras talvez prefiram servidores de diretórios dedicados, cuja única tarefa seja administrar certificados X.509. Tais diretórios poderiam fornecer serviços de pesquisa usando propriedades dos nomes X.500. Por exemplo, na teoria tal serviço de diretório poderia transferir uma consulta como: 'Forneça uma lista de todas as pessoas chamadas Alice que trabalham em departamentos de vendas de algum lugar nos Estados Unidos ou no Canadá'.

Revogação

O mundo real também está repleto de certificados, como de passaportes e carteiras de habilitação. Às vezes, esses certificados podem ser revogados, bem como as carteiras de habilitação para os que são flagrados dirigindo alcoolizados ou cometendo outras infrações de trânsito. O mesmo problema ocorre no mundo digital: a autoridade que concede um certificado pode decidir revogá-lo porque a pessoa ou organização que o possui cometeu algum abuso. Ele também pode ser revogado se a chave privada foi exposta ou, pior ainda, se a chave privada da CA foi comprometida. Desse modo, uma PKI precisa lidar com a questão da revogação. A possibilidade de revogação torna as coisas mais complicadas.

Um primeiro passo nessa direção é fazer cada CA emitir periodicamente uma lista de revogação de certificados, ou **CRL** (**Certificate Revocation List**), fornecendo os números de série de todos os certificados que ela revogou. Como os certificados contêm prazos de validade, a CRL só precisa conter os números de série de certificados ainda não vencidos. Uma vez que seu prazo de validade tenha passado, um certificado se torna automaticamente inválido, e assim não há necessidade de distinção entre os que alcançaram o prazo-limite e os que foram de fato revogados. Em ambos os casos, eles não podem mais ser utilizados.

Infelizmente, a introdução de CRLs significa que um usuário prestes a usar um certificado deve agora adquirir a CRL para ver se o certificado foi revogado. Se foi, ele não deve ser usado. Porém, mesmo que o certificado não esteja na lista, ele poderia ter sido revogado logo após a lista ter sido publicada. Desse modo, a única forma de realmente ter certeza é consultar a CA. Além disso, no próximo uso do certificado, a CA tem de ser consultada de novo, pois o certificado poderia ter sido revogado alguns segundos antes.

Outra complicação é o fato de um certificado revogado poder ser reabilitado, por exemplo, se tiver sido revogado pelo não pagamento de alguma taxa que foi paga posteriormente. Ter de lidar com a revogação (e talvez com a reabilitação) elimina uma das melhores propriedades dos certificados, ou seja, a possibilidade de usá-los sem ter de entrar em contato com uma CA.

Onde as CRLs devem ser armazenadas? Um boa alternativa seria armazená-las no mesmo local em que estão os próprios certificados. Uma estratégia é a CA publicar ativamente CRLs periódicas e fazer com que os diretórios as processem apenas removendo os certificados revogados. Se os diretórios não forem usados para armazenar os certificados, as CRLs poderão ser guardadas em cache em diversos locais convenientes na rede. Como uma CRL também é um documento assinado, se ela for adulterada, essa ação poderá ser facilmente detectada.

Se os certificados tiverem uma longa duração, as CRLs também serão longas. Por exemplo, se os cartões de crédito forem válidos por cinco anos, o número de revogações pendentes será muito maior do que seria se fossem emitidos novos cartões a cada três meses. Um modo-padrão de lidar com CRLs longas é emitir uma lista mestra com pouca frequência, mas emitir atualizações frequentes para a lista. Isso reduz a largura de banda necessária para distribuir as CRLs.

8.6 Segurança da comunicação

Agora, concluímos nosso estudo das principais ferramentas. A maior parte das técnicas e protocolos importantes foi abordada. O restante do capítulo estuda a aplicação dessas técnicas na prática para proporcionar segurança às redes, além de alguns conceitos sobre os aspectos sociais da segurança, no final do capítulo.

Nas quatro seções a seguir, examinaremos a segurança da comunicação, isto é, como levar os bits secretamente e sem alteração da origem até o destino, e como manter bits indesejáveis do lado de fora. Essas não são de modo algum as únicas questões de segurança em redes, mas certamente estão entre as mais importantes, o que nos dá um bom ponto de partida.

8.6.1 IPsec

A IETF sabia fazia muitos anos que havia carência de segurança na Internet. Não era fácil aumentá-la, porque havia uma disputa para definir onde colocá-la. A maioria dos especialistas em segurança acredita que, para serem realmente seguras, a criptografia e as verificações de integridade devem ser realizadas fim a fim (isto é, na camada de aplicação). Desse modo, o processo de origem criptografa e/ou protege a integridade dos dados e os envia ao processo de destino, onde eles são descriptografados e/ou verificados. Qualquer adulteração realizada entre esses dois processos, inclusive dentro de qualquer sistema operacional, poderá então ser detectada. A dificuldade com essa abordagem é que ela exige a troca de todas as aplicações, a fim de torná-las cientes da segurança. Nessa visão, a segunda melhor abordagem é inserir a criptografia na camada de transporte ou em uma nova camada entre a camada de aplicação e a camada de transporte, tornando-a ainda fim a fim, mas sem exigir que as aplicações sejam alteradas.

A visão oposta é que os usuários não entendem de segurança e não serão capazes de usá-la corretamente e, como ninguém quer modificar programas existentes, a camada de rede devia autenticar e/ou codificar pacotes sem que os usuários estejam envolvidos. Depois de anos de batalhas, essa visão ganhou apoio suficiente para que fosse definido um padrão de segurança da camada de rede. Em parte, o argumento era que realizar a codificação na camada de rede não impediria que usuários conscientes da segurança a implementassem na camada de aplicação e, até certo ponto, isso também poderia ajudar os usuários sem consciência da segurança.

O resultado dessa guerra foi um projeto chamado **IPsec** (**IP security**), descrito nas RFCs 2401, 2402 e 2406, entre outras. Nem todos os usuários desejam a criptografia (porque ela é dispendiosa em termos computacionais). Em vez de torná-la opcional, decidiu-se exigir a criptografia o tempo todo, mas permitir o uso de um algoritmo nulo. O algoritmo nulo é descrito e elogiado por sua simplicidade, facilidade de implementação e grande velocidade na RFC 2410.

O projeto completo do IPsec é uma estrutura para vários serviços, algoritmos e detalhamentos. A razão para vários serviços é que nem todo mundo quer pagar o preço de ter todos os serviços o tempo todo, e assim os serviços estão disponíveis à escolha de cada usuário. Os principais serviços são sigilo, integridade de dados e proteção contra ataques de reprodução (em que o intruso reproduz uma conver-

sação). Todos esses serviços se baseiam na criptografia de chave simétrica, porque o alto desempenho é fundamental.

A razão de vários algoritmos é que um algoritmo que agora é considerado seguro poderá ser violado no futuro. Tornar o IPsec independente do algoritmo mantém a estrutura mesmo se algum algoritmo específico for violado posteriormente.

A razão para vários níveis de detalhamento é possibilitar a proteção de uma única conexão TCP, de todo o tráfego entre um par de hosts ou de todo o tráfego entre um par de roteadores seguros, além de outras possibilidades.

Um aspecto um tanto surpreendente do IPsec é que, embora esteja na camada IP, ele é orientado a conexões. Na realidade, isso não é muito surpreendente porque, para ter alguma segurança, uma chave tem de ser estabelecida e usada por um período de tempo — basicamente, uma espécie de conexão com um nome diferente. Além disso, as conexões amortizam os custos de configuração por vários pacotes. Uma "conexão" no contexto do IPsec é chamada associação de segurança, ou **SA** (**Security Association**). Uma SA é uma conexão simplex entre duas extremidades e tem um identificador de segurança associado a ela. Se houver necessidade de tráfego seguro em ambos os sentidos, serão exigidas duas associações de segurança. Os identificadores de segurança são transportados em pacotes que percorrem essas conexões seguras e são usados para pesquisar chaves e outras informações relevantes ao chegar um pacote seguro.

Tecnicamente, o IPsec tem duas partes principais. A primeira parte descreve dois novos cabeçalhos que podem ser acrescentados aos pacotes para transportar o identificador de segurança, dados de controle de integridade e outras informações. A outra parte, **ISAKMP** (**Internet Security Association and Key Management Protocol**), trata do estabelecimento de chaves. ISAKMP é um framework. O protocolo principal para executar o trabalho é **IKE (Internet Key Exchange)**. Deve-se usar a versão 2 do IKE, descrita na RFC 4306, pois a versão anterior continha muitas falhas, conforme indicado por Perlman e Kaufman (2000).

O IPsec pode ser usado de dois modos. No **modo de transporte**, o cabeçalho IPsec é inserido logo após o cabeçalho IP. O campo *Protocolo* no cabeçalho IP é alterado para indicar que o cabeçalho IPsec vem após o cabeçalho IP normal (antes do cabeçalho TCP). O cabeçalho IPsec contém informações de segurança, principalmente o identificador de SA, um novo número de sequência e possivelmente uma verificação de integridade da carga útil.

No modo **tunelamento**, todo o pacote IP, incluindo o cabeçalho, é encapsulado no corpo de um novo pacote IP com um cabeçalho IP completamente novo. O modo túnel é útil quando o túnel termina em um local diferente do destino final. Em alguns casos, o fim do túnel é uma máquina com gateway de segurança, por exemplo, um firewall da empresa. Nesse modo, o gateway de segurança encapsula e desencapsula pacotes à medida que eles passam por ele. Quando o túnel termina nessa máquina segura, as máquinas da LAN da empresa não têm de tomar conhecimento do IPsec. Isso é tarefa do firewall.

O modo túnel também é útil quando um conjunto de conexões TCP é agregado e tratado como um único fluxo codificado, porque isso evita que um intruso veja quem está enviando, quem está recebendo e quantos pacotes são enviados. Às vezes, o simples conhecimento da quantidade de tráfego e de seu destino é uma informação valiosa. Por exemplo, se durante uma crise militar o volume de tráfego que flui entre o Pentágono e a Casa Branca cair de forma abrupta, mas o volume de tráfego entre o Pentágono e alguma instalação militar nas profundezas das Montanhas Rochosas do Colorado aumentar na mesma proporção, um intruso poderá deduzir algumas informações úteis desses dados. O estudo dos padrões de fluxo de pacotes, ainda que eles estejam codificados, é chamado **análise de tráfego**. O modo túnel fornece um meio para anular até certo ponto essa análise. A desvantagem do modo túnel é que ele acrescenta um cabeçalho IP extra, aumentando assim substancialmente o tamanho dos pacotes. Em contraste, o modo de transporte não afeta muito o tamanho dos pacotes.

O primeiro cabeçalho novo é o cabeçalho de autenticação, ou **AH (Authentication Header)**. Ele fornece verificação de integridade e segurança contra reprodução, mas não oferece sigilo (isto é, não há criptografia de dados). O uso do AH no modo de transporte é ilustrado na Figura 8.23. No IPv4, ele é inserido entre o cabeçalho IP (incluindo quaisquer opções) e o cabeçalho TCP. No IPv6 ele é simplesmente outro cabeçalho de extensão e é tratado como tal. De fato, o formato é próximo ao de um cabeçalho de extensão padrão do IPv6. É possível que a carga útil tenha de ser preenchida até completar algum tamanho específico para o algoritmo de autenticação, como mostra a figura.

Agora, vamos examinar o cabeçalho AH. O campo *Próximo cabeçalho* é usado para armazenar o valor anterior que o campo *Protocolo* do IP tinha antes de ser substituído por 51 para indicar que haverá um cabeçalho AH em seguida. Na maioria dos casos, o código para o TCP (6) ficará aqui. O campo *Tamanho da carga útil* é o número de palavras de 32 bits no cabeçalho AH, menos 2 unidades.

O campo *Índice de parâmetros de segurança* é o identificador da conexão. Ele é inserido pelo transmissor para indicar um registro específico no banco de dados do receptor. Esse registro contém a chave compartilhada usada nessa conexão e outras informações sobre a conexão. Se esse protocolo tivesse sido criado pela ITU e não pela IETF, esse campo seria chamado *Número do circuito virtual*.

O campo *Número de sequência* é usado para numerar todos os pacotes enviados em uma SA. Todo pacote recebe um número exclusivo, até mesmo as retransmissões. Em outras palavras, a retransmissão de um pacote recebe um número diferente do original (embora seu número de sequência do

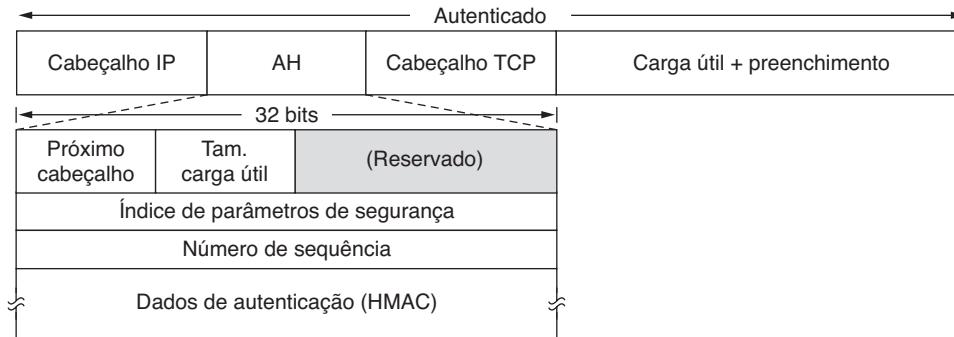



Figura 8.23 | O cabeçalho de autenticação do IPsec em modo de transporte para o IPv4.

TCP seja o mesmo). A finalidade desse campo é detectar ataques de reprodução. Esses números de sequência não podem se repetir. Se todos os 2^{32} se esgotarem, terá de ser estabelecida uma nova SA para dar continuidade à comunicação.

Finalmente, chegamos ao campo *Dados de autenticação*, um campo de comprimento variável, que contém a assinatura digital da carga útil. Quando a SA é estabelecida, os dois lados negociam o algoritmo de assinatura que vão usar. Normalmente, não é utilizada aqui a criptografia de chave pública, porque os pacotes devem ser processados de forma extremamente rápida e todos os algoritmos de chave pública conhecidos são lentos demais. Como o IPsec se baseia na criptografia de chave simétrica, e como o transmissor e o receptor negociam uma chave compartilhada antes de estabelecer uma SA, a chave compartilhada é usada no cálculo da assinatura. Um modo simples é calcular o hash sobre o pacote, somado à chave compartilhada. É claro que a chave compartilhada não é transmitida. Um esquema como esse é chamado **HMAC** (**Hashed Message Authentication Code**). É muito mais rápido calcular o valor desse esquema que executar primeiro o SHA-1 e depois executar o RSA sobre o resultado.

O cabeçalho AH não permite criptografia dos dados; portanto, ele é útil principalmente quando a verificação da integridade é necessária, mas não o sigilo. Uma característica do AH que vale a pena notar é que a verificação de integridade abrange alguns dos campos do cabeçalho IP, ou seja, aqueles que não se alteram à medida que o pacote passa de um roteador para outro. Por exemplo, o campo *Tempo de vida* muda a cada hop e assim não pode ser incluído na verificação de integridade. Contudo, o endereço de origem IP é incluído na verificação, o que torna impossível para um intruso falsificar a origem de um pacote.

O cabeçalho IPsec alternativo é a **ESP** (**Encapsulating Security Payload**). Seu uso no modo de transporte e no modo túnel é mostrado na Figura 8.24.

O cabeçalho ESP consiste em duas palavras de 32 bits. Elas constituem os campos *Índice de parâmetros de segurança* e *Número de sequência*, que vimos no AH. Uma terceira palavra que geralmente segue esses campos (mas tecnicamente não faz parte do cabeçalho) é o *Vetor de referência*, usado para a criptografia de dados, a menos que não seja utilizada criptografia, e nesse caso ele será omitido.

A ESP também fornece verificações de integridade do HMAC, como o AH; porém, em vez de ser incluídas no cabeçalho, elas vêm depois da carga útil, como mostra a Figura 8.24. A colocação do HMAC no final tem uma vantagem em uma implementação de hardware: o HMAC pode ser calculado à medida que os bits saem pela interface de rede e são acrescentados ao final. Por essa razão, as redes

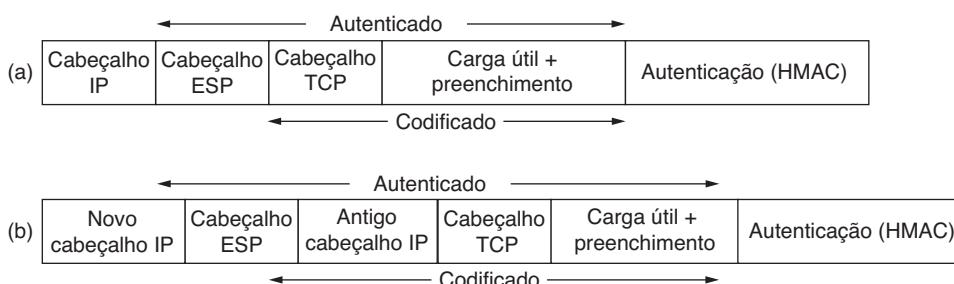



Figura 8.24 | (a) ESP em modo de transporte. (b) ESP em modo túnel.

Ethernet e outras LANs têm seus CRCs em um final (trailer), e não em um cabeçalho. Com o AH, o pacote tem de ser armazenado no buffer e a assinatura deve ser calculada antes que seja possível enviar o pacote, reduzindo potencialmente o número de pacotes/s que podem ser enviados.

Considerando que a ESP pode fazer tudo o que o AH pode fazer e muito mais, além de ser mais eficiente durante a fase inicial, surge a questão: afinal, qual é a necessidade do AH? A resposta é principalmente histórica. No início, o AH cuidava apenas da integridade, enquanto a ESP tratava do sigilo. Mais tarde, a integridade foi acrescentada à ESP, mas as pessoas que projetaram o AH não queriam deixá-lo morrer depois de tanto trabalho. No entanto, o único argumento real dessas pessoas se baseava no fato de que o AH é capaz de verificar uma parte do cabeçalho IP, o que a ESP não faz. Porém, esse é um argumento fraco, como também o argumento de que um produto com suporte para o AH, mas não para a ESP, poderia ter menos problemas para obter uma licença de exportação, porque não poderia efetuar a codificação. É provável que o AH fique defasado no futuro.

8.6.2 Firewalls

A capacidade de conectar um computador em qualquer lugar a outro computador em qualquer lugar é uma faca de dois gumes. Para as pessoas que estão em casa, é muito divertido navegar pela Internet. Para os gerentes de segurança das empresas, trata-se de um pesadelo. Muitas empresas têm grandes quantidades de informações confidenciais on-line — segredos comerciais, planos de desenvolvimento de produtos, estratégias de marketing, análises financeiras etc. A revelação dessas informações para um concorrente poderia ter consequências terríveis.

Além do perigo das informações virem a público, também há o perigo do vazamento dessas informações dentro da empresa. Em particular, vírus, worms e outras pestes digitais podem burlar a segurança, destruir dados valiosos e consumir muito tempo dos administradores que tentam eliminar a confusão causada por eles. Com frequência, eles são trazidos por funcionários descuidados que querem brincar com algum jogo novo muito divertido. Em consequência disso, são necessários mecanismos para manter os 'bons' bits e descartar os 'maus' bits. Um dos métodos é usar o IPsec, que protege os dados em trânsito entre sites seguros. No entanto, o IPsec não faz nada para impedir as pragas digitais nem os intrusos de invadir a LAN da empresa. Para ver como alcançar esse objetivo, precisamos examinar os firewalls.

Os **firewalls** são apenas uma adaptação moderna de uma antiga forma de segurança medieval: cavar um fosso profundo em torno do castelo. Esse recurso forçava todos aqueles que quisessem entrar ou sair do castelo a passar por uma única ponte levadiça, onde poderiam ser revistados por guardas. Nas redes, é possível usar o mesmo artifício: uma empresa pode ter muitas LANs conectadas de forma arbitrária, mas todo o tráfego de saída ou de entrada da empresa é feito através de uma ponte levadiça eletrônica (firewall), como mostra a Figura 8.25. Não existe outra rota.

O firewall atua como um **filtro de pacotes**. Ele inspeciona todo e qualquer pacote que entra e que sai. Os pacotes que atenderem a algum critério descrito nas regras formuladas pelo administrador da rede serão remetidos normalmente, mas os que falharem no teste serão descartados sem cerimônia.

O critério de filtragem normalmente é dado como regras ou em tabelas que listam as origens e os destinos aceitáveis, as origens ou destinos bloqueados e as regras-padrão que orientam o que deve ser feito com os pacotes recebidos de outras máquinas ou destinados a elas. No caso comum de uma configuração TCP/IP, uma origem ou destino consiste em uma porta e um endereço IP. As portas indicam qual é o serviço desejado. Por exemplo, a porta 25 do TCP é para correio eletrônico, a porta 80 é para HTTP. Algumas portas podem simplesmente ser bloqueadas. Por exemplo, uma empresa poderia bloquear os pacotes recebidos em relação a todos os endereços IP combinados com a porta 79 do TCP. Antigamente, era comum usar o serviço

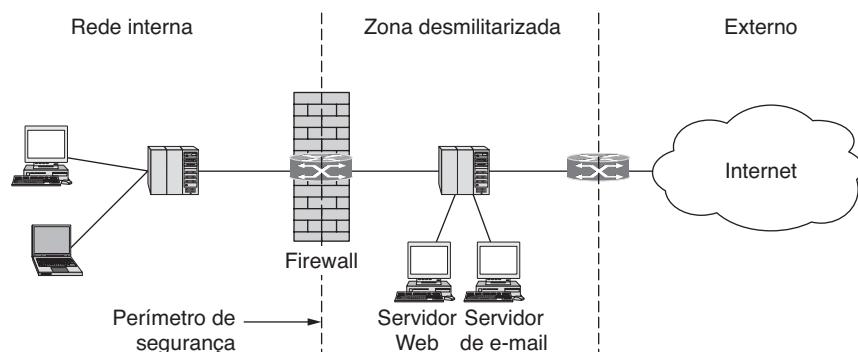



Figura 8.25 | Um firewall protegendo uma rede interna.

Finger para procurar os endereços de e-mail das pessoas, mas quase não é mais usado atualmente.

Outras portas não são bloqueadas com tanta facilidade. A dificuldade é que os administradores da rede desejam segurança, mas não podem cortar a comunicação com o mundo exterior. Esse esquema seria muito mais simples e melhor para a segurança, mas haveria infinitas reclamações por parte dos usuários. É para isso que existem esquemas como a zona desmilitarizada, ou **DMZ** (**DeMilitarized Zone**), mostrada na Figura 8.25. A DMZ é a parte da rede da empresa que se encontra fora do perímetro de segurança. Tudo passa por ali. Colocando uma máquina como um servidor Web na DMZ, os computadores na Internet podem fazer contato com ela para navegar pelo site Web da empresa. Agora, o firewall pode ser configurado para impedir o tráfego TCP que entra na porta 80, para que os computadores na Internet não possam usar essa porta para atacar os computadores na rede interna. Para permitir que o servidor seja administrado, o firewall pode ter uma regra para assegurar conexões entre as máquinas internas e o servidor Web.

Os firewalls se tornaram muito mais sofisticados com o tempo, em uma corrida contra os invasores. Originalmente, os firewalls aplicavam um conjunto de regras independentes para cada pacote, mas se tornou difícil escrever regras que permitissem a funcionalidade e impedissem todo o tráfego indesejado. **Firewalls em estado de conexão** mapeiam os pacotes para conexões e usam campos do cabeçalho TCP/IP para cuidar da conectividade. Isso permite o uso de regras que, por exemplo, possibilitam que um servidor Web externo envie pacotes para um host interno, mas somente se o host interno primeiro estabelecer uma conexão com o servidor Web externo. Essa regra não é possível em redes que não mantêm o estado das conexões , que precisam passar ou descartar todos os pacotes vindos do servidor Web externo.

Outro nível de sofisticação possível com o processamento em estado de conexão é que o firewall implemente **gateways em nível de aplicação**. Esse processamento envolve o firewall examinando dentro dos pacotes, mesmo além do cabeçalho TCP, para ver o que a aplicação está fazendo. Com essa capacidade, é possível distinguir entre o tráfego HTTP usado para navegação Web e o tráfego HTTP usado para compartilhamento de arquivos peer-to-peer. Os administradores podem escrever regras para livrar a empresa do compartilhamento de arquivos peer-to-peer, mas permitir a navegação Web, que é vital para os negócios. Para todos esses métodos, o tráfego de saída pode ser inspecionado assim como o tráfego de entrada, por exemplo, para impedir que documentos confidenciais sejam remetidos para fora da empresa por correio eletrônico.

Como a discussão anterior deve deixar claro, os firewalls violam a disposição de camadas do padrão de protocolos. Eles são dispositivos da camada de rede, mas, para realizar sua filtragem, examinam as camadas de transporte e aplicação. Isso os torna frágeis. Por exemplo, os firewalls costumam contar com as convenções-padrão de numeração de porta para determinar o tipo de tráfego transportado em um pacote. As portas-padrão normalmente são usadas, mas não por todos os computadores nem por todas as aplicações. Algumas aplicações peer-to-peer selecionam portas dinamicamente, para evitar que sejam facilmente localizadas (e bloqueadas). A codificação com IPsec ou outros esquemas esconde do firewall as informações da camada mais alta. Finalmente, um firewall não pode falar prontamente com os computadores que se comunicam através dele para informar quais diretrizes estão sendo aplicadas e por que sua conexão está sendo descartada. Ele deve simplesmente fingir ser um fio partido. Por todas essas razões, os que se preocupam com a pureza das redes consideram os firewalls uma mancha na arquitetura da Internet. Porém, a Internet pode ser um local perigoso se você é um computador. Os firewalls ajudam nesse aspecto, e por isso provavelmente permanecerão.

Mesmo que o firewall esteja perfeitamente configurado, ainda existem vários problemas de segurança. Por exemplo, se um firewall estiver configurado para permitir apenas a entrada de pacotes de redes específicas (por exemplo, outras instalações da empresa), um intruso fora do firewall pode inserir falsos endereços de origem para ultrapassar essa verificação. Se um usuário interno quiser transportar documentos secretos para fora da empresa, ele poderá codificar ou até mesmo fotografar os documentos e transportar as fotografias como arquivos JPEG, que conseguirão passar por quaisquer filtros de correio. Não discutimos nem mesmo o fato de que, embora três quartos de todos os ataques venham de fora do firewall, os ataques que vêm de dentro do firewall (por exemplo, de funcionários insatisfeitos) normalmente são os mais perigosos (Verizon, 2009).

Um problema diferente com os firewalls é que eles oferecem um único perímetro de defesa. Se essa defesa for rompida, tudo estará perdido. Por esse motivo, costumam ser usados em uma defesa em camadas. Por exemplo, um firewall pode proteger a entrada para a rede interna e cada computador também pode ter seu próprio firewall. Os leitores que acharem que um ponto de verificação de segurança é suficiente certamente nenhum voo internacional em uma companhia aérea regular recentemente.

Além disso, há toda uma classe de diferentes ataques com que os firewalls não podem lidar. A ideia básica por trás de um firewall é impedir a entrada de intrusos e a saída de dados secretos. Infelizmente, existem pessoas que não têm nada melhor para fazer do que tentar derrubar certos sites. Para isso, elas enviam ao destino pacotes legítimos em grande quantidade, até o site entrar em colapso com a carga. Por exemplo, para incapacitar um site Web, um intruso pode enviar um pacote *SYN* do TCP para estabelecer uma conexão. Então, o site alocará um slot da tabela para a conexão e enviará um pacote $SYN + ACK$ como resposta. Se o intruso não responder, o slot da tabela ficará retido por alguns segundos até um tempo-limite. Se o intruso enviar milhares de solicitações de conexão, todos os slots da tabela serão preenchidos

e nenhuma conexão legítima poderá passar. Os ataques em que o objetivo do intruso é desativar o destino em vez de roubar dados são chamados ataques de negação de serviço, ou **DoS** (**Denial of Service**). Em geral, os pacotes solicitados têm endereços de origem falsos, para que o intruso não possa ser rastreado com facilidade. Os ataques de DoS contra sites importantes são comuns na Internet.

Uma variante ainda pior é aquela em que o intruso já entrou em centenas de computadores em outros lugares do mundo, e depois comanda todos esses computadores em um ataque ao mesmo alvo ao mesmo tempo. Essa estratégia não apenas aumenta o poder de fogo do intruso, mas também reduz a chance de detecção, pois os pacotes estão vindo de um grande número de máquinas pertencentes a usuários insuspeitos. Um ataque desse tipo é chamado **DDoS (Distributed Denial of Service)**, e é muito difícil proteger-se contra ele. Ainda que a máquina atacada possa reconhecer rapidamente uma solicitação falsa, processar e descartar a solicitação é uma operação que leva algum tempo e, se chegarem solicitações em número suficiente por segundo, a CPU passará todo seu tempo lidando com elas.

8.6.3 Redes privadas virtuais

Muitas empresas têm escritórios e fábricas espalhados por muitas cidades, às vezes por vários países. Antigamente, antes das redes públicas de dados, era comum tais empresas arrendarem linhas dedicadas da companhia telefônica entre alguns pares de locais ou entre todos eles. Algumas empresas ainda fazem isso. Uma rede construída a partir de computadores de empresas e de linhas telefônicas dedicadas é chamada **rede privada**.

As redes privadas funcionam muito bem e são bastante seguras. Se as únicas linhas disponíveis forem as linhas dedicadas, nenhum tráfego poderá vazar para fora das instalações da empresa, e os intrusos terão de grampear fisicamente as linhas para entrar, o que não é fácil. O problema das redes privadas é que arrendar uma única linha T1 custa milhares de dólares por mês, e as linhas T3 têm um custo muito elevado. Quando surgiram as redes públicas de dados e mais tarde a Internet, muitas empresas optaram por mover seu tráfego de dados (e possivelmente o de voz) para a rede pública, mas sem desistir da segurança da rede privada.

Essa demanda logo levou à criação de redes privadas virtuais, ou **VPNs** (**Virtual Private Networks**), que são redes sobrepostas às redes públicas, mas com a maioria das propriedades das redes privadas. Elas são chamadas 'virtuais' porque são meramente uma ilusão, da mesma forma que os circuitos virtuais não são circuitos reais e que a memória virtual não é uma memória real.

Uma técnica popular é construir as VPNs diretamente sobre a Internet. Um projeto comum é equipar cada escritório com um firewall e criar túneis pela Internet entre todos os pares de escritórios, como ilustra a Figura 8.26(a). Outra vantagem do uso da Internet para a conectividade é que os túneis podem ser criados por demanda para incluir, por exemplo, o computador de um funcionário que está em casa ou viajando, desde que a pessoa tenha uma conexão com a Internet. Essa flexibilidade é muito maior do que aquela oferecida com linhas dedicadas, embora, do ponto de vista dos computadores na VPN, a topologia seja exatamente a mesma que no caso da rede privada, como mostra a Figura 8.26(b). Quando o sistema é criado, cada par de firewalls tem de negociar os parâmetros de sua SA, incluindo os serviços, os modos, os algoritmos e as chaves. Se o IPsec for usado no tunelamento, será possível agregar todo o tráfego entre dois pares de escritórios quaisquer em uma única SA autenticada e criptografada, fornecendo assim controle de integridade, sigilo e até mesmo uma considerável imunidade à analise de tráfego. Muitos firewalls têm recursos internos para VPN. Alguns roteadores comuns podem fazer isso muito bem, porém, como os firewalls se destinam principalmente a questões de segurança, é natural fazer os túneis começar e terminar nos firewalls, proporcionando uma separação clara entre a empresa e a Internet. Desse modo, firewalls, VPNs e IPsec com ESP em modo túnel formam uma combinação natural e muito utilizada na prática.

Depois que as SAs são estabelecidas, o tráfego pode começar a fluir. Para um roteador na Internet, um pacote que viaja por um túnel VPN é apenas um pacote comum. O único detalhe incomum com ele é a presença do cabe-

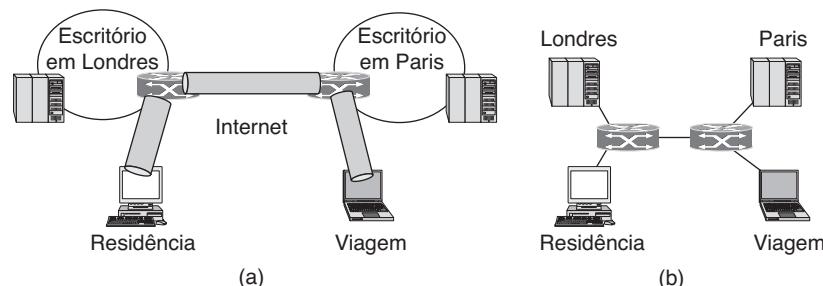



Figura 8.26 | (a) Uma rede privada virtual. (b) Topologia vista de dentro.

çalho IPsec depois do cabeçalho IP; porém, como esses cabeçalhos extras não têm nenhum efeito sobre o processo de encaminhamento, os roteadores não se preocupam com esse cabeçalho extra.

Outra técnica que está ganhando popularidade é fazer com que o ISP crie a VPN. Usando MPLS (conforme discutimos no Capítulo 5), os caminhos para o tráfego da VPN podem ser criados pela rede do ISP entre os escritórios da empresa. Esses caminhos mantêm o tráfego da VPN separado do restante do tráfego da Internet e podem receber alguma quantidade de largura de banda garantida, ou então outro tipo de qualidade de serviço.

Uma vantagem importante dessa forma de organizar uma VPN é sua completa transparência para todo o software do usuário. Os firewalls configuram e gerenciam as SAs. A única pessoa consciente dessa configuração é o administrador do sistema, que tem de configurar e administrar os gateways de segurança, ou o administrador do ISP, que tem de configurar os caminhos MPLS. Para todas as outras pessoas, é como ter de novo uma rede privada de linha dedicada. Para obter mais informações sobre VPNs, consulte Lewis (2006).

8.6.4 Segurança em redes sem fios

É fácil projetar um sistema com total segurança em termos lógicos usando VPNs e firewalls, muito embora na prática ele vaze como uma peneira. Essa situação pode ocorrer se algumas das máquinas forem sem fios e usarem comunicação por rádio, que passa pelo firewall em ambos os sentidos (entrada e saída). O alcance das redes 802.11 frequentemente é de algumas centenas de metros; assim, qualquer pessoa que queira espionar uma empresa pode se dirigir até o estacionamento dos funcionários pela manhã, deixar um notebook capaz de reconhecer sinais 802.11 dentro do carro para registrar tudo o que ouvir e se retirar no final do dia. À tarde, o disco rígido estará repleto de valiosas informações. Teoricamente, esse vazamento não deveria acontecer. Na teoria, as pessoas também não deveriam roubar bancos.

Grande parte do problema de segurança pode ter sua origem nos fabricantes de estações-base sem fios (pontos de acesso) que tentam tornar seus produtos amigáveis para o usuário. Em geral, se o usuário retirar o dispositivo da caixa e o conectar à tomada da rede elétrica, ele começará a operar de imediato — quase sempre sem nenhuma segurança, revelando segredos para todo mundo que estiver dentro do alcance de rádio. Se ele for conectado a uma rede Ethernet, todo tráfego da Ethernet também aparecerá de repente no estacionamento. A rede sem fios é um sonho que se tornou realidade para o espião: dados gratuitos sem nenhum trabalho. Por isso, não é preciso dizer que a segurança é ainda mais importante para sistemas sem fios que para sistemas fisicamente conectados. Nesta seção, examinaremos alguns aspectos de segurança das redes sem fios. Algumas informações adicionais podem ser encontradas em Nichols e Lekkas (2002).

Segurança de redes 802.11

Parte do padrão 802.11, originalmente chamado **802.11i**, prescreve um protocolo de segurança do nível de enlace de dados para impedir que um nó sem fios leia ou interfira nas mensagens enviadas entre outro par de nós sem fios. Ele também é chamado de **WPA2 (WiFi Protected Access 2)**. O WPA puro é um esquema intermediário que implementa um subconjunto do 802.11i. Ele deve ser evitado em favor do WPA2.

Vamos descrever o 802.11i em breve, mas primeiro indicaremos que ele é um substituto para o **WEP** (**Wired Equivalent Privacy**), a primeira geração de protocolos de segurança 802.11. O WEP foi criado por um comitê de padrões de rede, que é um processo completamente diferente, por exemplo, do modo como o NIST selecionou o projeto do AES. Os resultados foram devastadores. O que saiu errado com ele? Quase tudo, do ponto de vista da segurança. Por exemplo, WEP codificava dados por confidencialidade realizando um XOR com a saída de um fluxo de cifras. Infelizmente, arrumações de chave fracas fizeram com que a saída geralmente fosse reutilizada. Isso ocasionou formas triviais de ataque. Como outro exemplo, a verificação de integridade era baseada em um CRC de 32 bits. Esse é um código eficiente para detectar erros de transmissão, mas não é um mecanismo criptograficamente forte para combater os invasores.

Essas e outras falhas de projeto tornaram o WEP muito fácil de ser comprometido. A primeira demonstração prática de que o WEP tinha falhas veio quando Adam Stubblefield era um estagiário na AT&T (Stubblefield et al., 2002). Ele conseguiu codificar e testar um ataque esboçado por Fluhrer et al. (2001) em uma semana, em que a maior parte do tempo foi gasta convencendo a gerência para que lhe comprasse uma placa WiFi para usar em seus experimentos. O software para quebrar senhas WEP em um minuto agora está livremente disponível e o uso de WEP é bastante desencorajado. Embora ele impeça o acesso casual, não oferece nenhuma forma de segurança real. O grupo 802.11i foi reunido às pressas quando ficou claro que o WEP estava seriamente em perigo. Esse grupo produziu um padrão formal em junho de 2004.

Agora, vamos descrever o 802.11i, que oferece segurança real se for montado e usado devidamente. Existem dois cenários comuns em que o WPA2 é usado. O primeiro é um ambiente corporativo, em que uma empresa tem um servidor de autenticação separado, com um banco de dados de nomes de usuários e senhas, que pode ser usado para determinar se um cliente sem fios tem permissão para acessar a rede. Nesse ambiente, os clientes usam protocolos-padrão para ser autenticados na rede. Os principais padrões são o **802.1X**, com o qual o ponto de acesso permite que o cliente trave um diálogo com o servidor de autenticação e observe o resultado, e o **EAP** (**Extensible Authentication Protocol**) (RFC 3748), que informa como o cliente e o servidor de autenticação devem interagir. Na realidade,

EAP é um framework e outros padrões definem as mensagens do protocolo. Porém, não vamos entrar em muitos detalhes dessa troca, pois eles não são muito importantes para uma visão geral do assunto.

O segundo cenário está em um ambiente doméstico, em que não existe um servidor de autenticação. Em vez disso, há uma senha única compartilhada, usada pelos clientes para acessar a rede sem fios. Esse ambiente é menos complexo do que aquele com um servidor de autenticação, motivo pelo qual é usado em casa e em pequenas empresas, porém também é menos seguro. A principal diferença é que, com um servidor de autenticação, cada cliente recebe uma chave para codificar o tráfego, que não é conhecida pelos outros clientes. Com uma única senha compartilhada, diferentes chaves são derivadas para cada cliente, mas todos têm a mesma senha e podem derivar as chaves uns dos outros, se quiserem.

As chaves usadas para codificar o tráfego são calculadas como parte de um handshake de autenticação. O handshake ocorre logo depois que o cliente se associa a uma rede sem fios e se autentica com um servidor de autenticação, se houver. No início do handshake, o cliente tem a senha de rede compartilhada ou sua senha para o servidor de autenticação. Essa senha é usada para derivar uma chave mestra. Porém, a chave mestra não é usada diretamente para codificar pacotes. É uma prática criptográfica padrão derivar uma chave de sessão para cada período de uso, mudar a chave para diferentes sessões e expor a chave mestra à observação o mínimo possível. É essa chave de sessão que é calculada no handshake.

A chave de sessão é calculada com o handshake de quatro pacotes mostrado na Figura 8.27. Primeiro, o ponto de acesso, ou AP (Access Point), envia um número qualquer para identificação. Nos protocolos de segurança como esse, os números aleatórios, usados apenas uma vez, são chamados **nonces**, que é uma contração de '*number used once*' (número usado uma vez). O cliente também escolhe seu próprio nonce. Ele usa os nonces, seu endereço MAC e o do AP, e a chave mestra para calcular uma chave de sessão, K_S. A chave de sessão é dividida em partes, usadas para diferentes finalidades, mas omitiremos esse detalhe. Agora, o cliente tem chaves de sessão, mas o AP não tem. Assim, o cliente envia seu nonce ao AP, e este realiza o mesmo cálculo para derivar as mesmas chaves de sessão. Os nonces podem ser enviados abertamente, pois as chaves não podem ser derivadas a partir deles sem informações extras, secretas. A mensagem do cliente é protegida com uma verificação de integridade chamada verificação de integridade da mensagem, ou **MIC (Message Integrity Check**), com base na chave da sessão. O AP pode verificar se a MIC está correta, e portanto se a mensagem realmente veio do cliente, depois de calcular as chaves de sessão. Uma MIC é apenas outro nome para um código de autenticação de mensagem, como em um HMAC. Em seu lugar, o termo MIC é frequentemente utilizado para protocolos de rede, devido ao potencial de confusão com endereços MAC (Medium Access Control).

Nas duas últimas mensagens, o AP distribui uma chave de grupo, K_G, ao cliente, e este confirma a mensagem. O recebimento dessas mensagens permite que o cliente verifique se o AP tem as chaves de sessão corretas e vice-versa. A chave de grupo é usada para tráfego de broadcast e multicast na LAN 802.11. Tendo em vista que o resultado do handshake é que cada cliente tem suas próprias chaves de codificação, nenhuma delas pode ser usada pelo AP para transmitir pacotes por broadcast a todos os clientes sem fios; uma cópia separada teria de ser enviada a cada cliente usando sua chave. Em vez disso, uma chave compartilhada é distribuída para que o tráfego de broadcast possa ser enviado apenas uma vez e recebido por todos os clientes. Ela precisa ser atualizada à medida que os clientes entram e saem da rede.

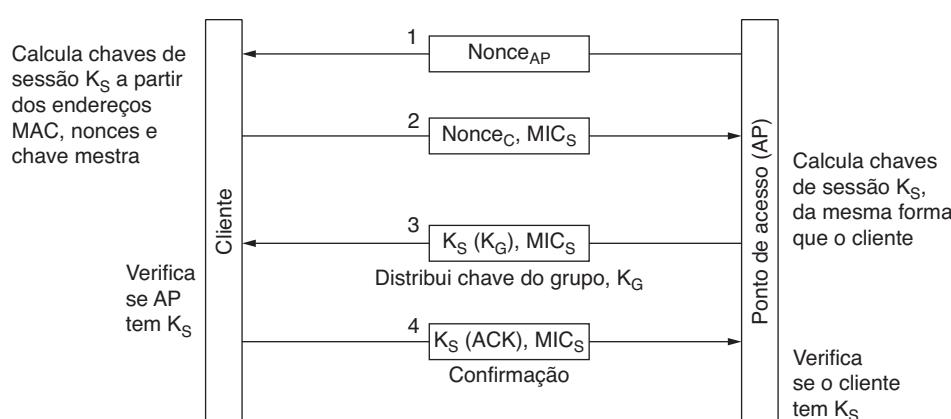



Figura 8.27 | O handshake de definição de chave no 802.11i.

Por fim, chegamos à parte em que as chaves são realmente usadas para fornecer segurança. Dois protocolos podem ser usados no 802.11i para fornecer confidencialidade, integridade e autenticação da mensagem. Assim como WPA, um dos protocolos, chamado **TKIP** (**Temporary Key Integrity Protocol**), foi uma solução temporária. Ele foi criado para melhorar a segurança em placas 802.11 antigas e lentas, de modo que pelo menos alguma segurança melhor que o WEP pudesse ser implementada como um upgrade no firmware. Contudo, ele também foi quebrado, e por isso é melhor ficar com o outro protocolo recomendado, o **CCMP**. O que significa CCMP? Essa é uma abreviação para o nome espetacular 'Counter mode with Cipher block chaining Message authentication code Protocol' (protocolo de código de autenticação de mensagem de modo contador com encadeamento de blocos de cifras). Vamos chamá-lo simplesmente de CCMP. Você pode chamá-lo como quiser.

O CCMP funciona de uma maneira relativamente simples. Ele usa a codificação AES com uma chave e um tamanho de bloco de 128 bits. A chave vem da chave de sessão. Para fornecer confidencialidade, as mensagens são codificadas com AES no modo contador. Lembre-se de que discutimos sobre os modos de cifras na Seção 8.2.3. Esses modos são o que impede que a mesma mensagem seja codificada para o mesmo conjunto de bits todas as vezes. O modo contador insere um contador junto com a codificação. Para oferecer integridade, a mensagem, incluindo os campos de cabeçalho, é codificada com o modo de encadeamento de blocos de cifras e o último bloco de 128 bits é mantido como o MIC. Em seguida, tanto a mensagem (codificada com o modo contador) como o MIC são enviados. O cliente e o AP podem realizar essa codificação individualmente, ou então verificar essa codificação quando um pacote for recebido pela rede sem fios. Para o broadcast ou multicast de mensagens, o mesmo procedimento é usado com a chave de grupo.

Segurança do Bluetooth

O Bluetooth tem um alcance bem mais curto que o 802.11, portanto não pode ser atacado do estacionamento, embora a segurança ainda seja uma questão importante nesse caso. Por exemplo, imagine que o computador de Alice esteja equipado com um teclado Bluetooth sem fio. Na ausência de segurança, se Trudy estiver no escritório ao lado, ela poderá ler tudo o que Alice digitou, inclusive todo e-mail enviado. Ela também poderá captar tudo que o computador de Alice enviar à impressora Bluetooth mais próxima (por exemplo, as mensagens de correio eletrônico recebidas e os relatórios confidenciais). Felizmente, o Bluetooth tem um esquema de segurança elaborado para tentar anular as Trudies desse mundo. Agora vamos resumir as principais características desse esquema.

A partir da versão 2.1, o Bluetooth tem quatro modos de segurança, variando desde nenhuma segurança até total criptografia de dados e controle de integridade. Como ocorre com o 802.11, se a segurança for desativada (o padrão para dispositivos mais antigos), não haverá nenhuma segurança. A maioria dos usuários mantém a segurança desativada até ocorrer uma séria violação; depois eles a ativam. No mundo agrícola, essa abordagem é conhecida como trancar a porteira depois que o cavalo escapou.

O Bluetooth fornece segurança em várias camadas. Na camada física, os saltos de frequência oferecem um nível mínimo de segurança, mas, como qualquer dispositivo Bluetooth que se desloca em uma piconet tem de ser informado da sequência de saltos de frequência, é óbvio que essa frequência não é um segredo. A segurança real começa quando o escravo recém-chegado solicita um canal ao mestre. Antes do Bluetooth 2.1, havia o pressuposto de que os dois dispositivos compartilhavam uma chave secreta configurada com antecedência. Em alguns casos, ambos são fisicamente conectados pelo fabricante (por exemplo, um fone de ouvido e um telefone celular vendidos como uma unidade). Em outros, um dispositivo (como o fone de ouvido) tem uma chave embutida no código e o usuário tem de digitar essa chave no outro dispositivo (o telefone celular) como um número decimal. Essas chaves compartilhadas são chamadas **passkeys** (ou chaves de passagem). Infelizmente, as passkeys costumam vir definidas como '1234' ou outro valor previsível, e de qualquer forma são quatro dígitos decimais, permitindo apenas 10^4 escolhas. Com o emparelhamento seguro simples do Bluetooth 2.1, os dispositivos escolhem um código a partir de uma faixa de seis dígitos, tornando a passkey muito menos previsível, mas ainda longe de ser segura.

Para estabelecer um canal, o escravo e o mestre verificam se a outra máquina conhece a passkey. Nesse caso, eles negociam para ver se esse canal será criptografado, terá sua integridade controlada ou ambos. Em seguida, selecionam uma chave de sessão aleatória de 128 bits, na qual alguns bits podem ser públicos. A razão para permitir o enfraquecimento dessa chave é obedecer a algumas restrições do governo de vários países, criadas para impedir a exportação ou o uso de chaves mais longas do que aquelas que o governo é capaz de violar.

A criptografia utiliza um fluxo de cifras chamado $\mathbf{E_0}$; o controle de integridade emprega o **SAFER+**. Ambos são blocos de cifras de chave simétrica tradicional. O SAFER+ foi submetido aos rígidos testes de aprovação do AES, mas foi eliminado na primeira rodada, por ser mais lento que os outros candidatos. O Bluetooth ficou pronto antes de ser escolhida a cifra do AES; caso contrário, é mais provável que ele tivesse usado o Rijndael.

A criptografia real usando um fluxo de cifras é mostrada na Figura 8.12, com o texto simples sendo submetido a um XOR com o fluxo de chaves para gerar o texto cifrado. Infelizmente, o próprio E_0 (como o RC4) pode ter deficiências fatais (Jakobsson e Wetzel, 2001). Embora ele não tenha sido rompido na época em que escrevemos, sua semelhança com a cifra A5/1, cuja falha espetacular compromete todo o tráfego de telefones GSM, causa preocupação (Biryukov et al., 2000). Às vezes, é espantoso perceber (até mesmo para os autores deste livro) que, no eterno jogo de

gato e rato entre criptógrafos e criptoanalistas, os criptoanalistas frequentemente sejam os vencedores.

Outra questão de segurança é que o Bluetooth autentica apenas dispositivos, não usuários; assim, o furto de um dispositivo Bluetooth pode dar ao ladrão acesso às finanças e às outras contas do usuário. No entanto, o Bluetooth também implementa segurança nas camadas superiores. Portanto, até mesmo na eventualidade de uma violação da segurança no nível de enlace, deve restar alguma segurança, especialmente para aplicações que exigem a digitação de um código PIN em algum tipo de teclado para completar a transação.

8.7 Protocolos de autenticação

A **autenticação** é a técnica pela qual um processo confirma que seu parceiro na comunicação é quem deve ser e não um impostor. Confirmar a identidade de um processo remoto, diante da presença de um intruso ativo mal-intencionado, é surpreendentemente difícil e exige protocolos complexos baseados em criptografia. Nesta seção, estudaremos alguns dos protocolos de autenticação usados em redes de computadores com falhas na segurança.

A propósito, algumas pessoas confundem autorização com autenticação. A autenticação visa a determinar se você está ou não se comunicando com um processo específico. A autorização se preocupa com o que esse processo tem permissão para fazer. Por exemplo, um processo cliente entra em contato com um servidor de arquivos e afirma: 'Sou o processo do Scott e quero excluir o arquivo *receitas.old*'. Do ponto de vista do servidor de arquivos, as seguintes perguntas devem ser respondidas:

1. Esse processo é realmente de Scott (autenticação)?
2. Scott tem permissão para excluir *receitas.old* (autorização)?

Somente depois que ambas as perguntas forem respondidas afirmativamente, sem nenhuma ambiguidade, a ação solicitada poderá ser executada. Na verdade, a primeira é a mais importante. Depois que o servidor de arquivos souber com quem está se comunicando, verificar a autorização é uma simples questão de pesquisar entradas de tabelas ou bancos de dados locais. Por isso, nesta seção vamos nos concentrar na autenticação.

O modelo genérico que todos os protocolos de autenticação utilizam é descrito a seguir. Alice começa enviando uma mensagem para Bob ou para um **KDC** (**Key Distribution Center**) no qual confia e que sempre é honesto. Acontecem muitas outras trocas de mensagens em diferentes sentidos. À medida que elas são enviadas, uma intrusa mal-intencionada, Trudy, pode interceptar, modificar ou reproduzir essas mensagens a fim de enganar Alice e Bob, ou apenas para atrapalhar.

Todavia, quando a execução do protocolo tiver sido concluída, Alice terá certeza de que está se comunicando com Bob e vice-versa. Além disso, na maioria dos protocolos, os dois também terão estabelecido uma **chave de sessão** secreta que deverá ser usada durante a conversação. Na prática, por motivos de desempenho, todo o tráfego de dados é criptografado utilizando-se o modo de chave simétrica (em geral, AES ou DES triplo) — embora a criptografia de chave pública seja muito utilizada nos próprios protocolos de autenticação e para estabelecer a chave de sessão.

O objetivo de utilizar uma nova chave de sessão, escolhida ao acaso para cada nova conexão, é minimizar o volume de tráfego provocado pelo envio das chaves secretas ou públicas dos usuários, reduzir o volume de texto cifrado que um intruso pode obter e minimizar os danos, caso haja uma pane em um processo e seu dump de memória caia em mãos erradas. É muito provável que a única chave presente seja a de sessão. Todas as chaves permanentes deverão ser cuidadosamente zeradas depois que a sessão for estabelecida.

8.7.1 Autenticação baseada em chave secreta compartilhada

Para nosso primeiro protocolo de autenticação, vamos supor que Alice e Bob já compartilham uma chave secreta, K_{AB}. Essa chave compartilhada pode ter sido definida pelos dois em uma conversa telefônica ou pessoalmente, mas não na rede (que não é segura).

Ele se baseia em um princípio encontrado em muitos protocolos de autenticação: um dos lados envia um número aleatório ao outro, que em seguida o transforma de algum modo especial e retorna o resultado. Tais protocolos são chamados protocolos de **desafio-resposta**. Nesse e nos próximos protocolos de autenticação, será usada a seguinte notação:

A e *B* são as identidades de Alice e Bob.
R_i's são desafios, e *i* identifica o desafiante
K_i's são chaves, onde *i* indica o proprietário
K_S é a chave da sessão

A sequência de mensagens de nosso primeiro protocolo de autenticação de chave compartilhada é mostrada na Figura 8.28. Na mensagem 1, Alice envia sua identidade *A* para Bob, de uma forma que Bob entenda. É claro que Bob não tem como saber se essa mensagem veio de Alice ou de Trudy; portanto, ele escolhe um desafio, um número aleatório muito extenso, R_B, e o envia de volta a 'Alice' como sua mensagem número 2 em texto simples. Em seguida, Alice criptografa a mensagem com a chave que compartilha com Bob e envia o texto cifrado, $K_{AB}(R_B)$, de volta na mensagem 3. Quando vê a mensagem, Bob sabe de imediato que ela veio de Alice, pois Trudy não conhece K_{AB} e, portanto, não poderia tê-la gerado. Além disso, como o número R_B foi escolhido ao acaso a partir de um espaço muito extenso (digamos, números aleatórios de 128 bits), é muito improvável que Trudy tenha visto R_B e sua resposta em uma sessão anterior. É também improvável que ela consiga adivinhar a resposta correta a qualquer desafio.

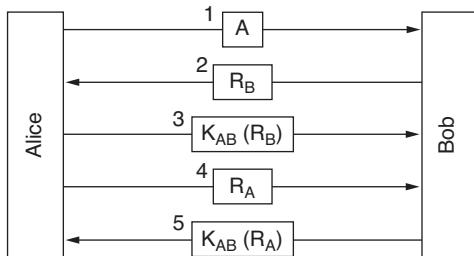
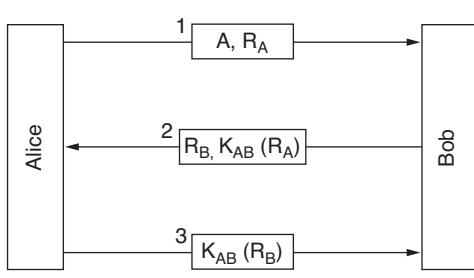



Figura 8.28 | Uma autenticação bidirecional utilizando um protocolo de desafio-resposta.

A essa altura, Bob tem certeza de que está se comunicando com Alice, mas Alice não tem certeza de nada, pois sabe que Trudy pode ter interceptado a mensagem 1 e enviado R_B como resposta. Talvez Bob tenha morrido na noite passada. Para descobrir com quem está se comunicando, Alice seleciona um número aleatório, R_A, e o envia a Bob como texto simples, na mensagem 4. Quando Bob responde com K_{AB} (R_A), Alice se certifica de que está se comunicando com Bob. Se eles quiserem estabelecer uma chave de sessão agora, Alice poderá selecionar uma, K_S, e enviá-la a Bob criptografada com K_{AB}.

O protocolo da Figura 8.28 contém cinco mensagens. Vamos ver se podemos eliminar algumas delas. Uma abordagem é ilustrada na Figura 8.29. Nessa figura, Alice inicia o protocolo de desafio-resposta em vez de esperar que Bob o faça. Da mesma forma, enquanto responde ao desafio de Alice, Bob envia o dele: o protocolo inteiro pode ser reduzido a três mensagens, em vez de cinco.

Esse novo protocolo representa um aperfeiçoamento em relação ao original? Em certo sentido, isso é verdade, pois agora o protocolo está mais curto. Infelizmente, também está errado. Sob determinadas circunstâncias, Trudy é capaz de enganar esse protocolo utilizando o método conhecido como **ataque por reflexão**. Isso quer dizer que Trudy poderá rompê-lo se for possível abrir várias sessões com Bob ao mesmo tempo. Por exemplo, essa situação seria verdadeira se Bob fosse um banco e estivesse preparado para aceitar muitas conexões simultâneas enviadas por caixas eletrônicos ao mesmo tempo.

O ataque por reflexão de Trudy é mostrado na Figura 8.30. Ele começa com Trudy afirmando ser Alice e enviando R_T. Bob responde, como sempre, com seu próprio desafio, R_B. Agora Trudy está em apuros. O que ela pode fazer? Ela não conhece K_{AB} (R_B).

Ela pode abrir outra sessão com a mensagem 3, fornecendo o R_B extraído da mensagem 2 como seu desafio. Bob o criptografa calmamente e envia K_{AB} (R_B) na mensagem 4. Representamos com sombreados as mensagens da segunda sessão, a fim de destacá-las. Agora, Trudy tem as informações que faltavam e, portanto, pode concluir a primeira sessão e abandonar a segunda. Nesse momento, Bob está convencido de que Trudy é Alice e, quando ela pede o saldo da conta, ele o informa sem mais perguntas. Em seguida, quando Trudy pede a Bob que transfira todo o dinheiro para uma conta secreta na Suíça, ele não hesita em fazê-lo.

A moral da história é a seguinte:

Projetar um protocolo de autenticação correto é mais difícil do que parece.

As quatro regras gerais a seguir frequentemente ajudam o projetista a evitar as armadilhas mais comuns:

1. fazer que o transmissor prove quem é antes de o receptor responder. Nesse caso, Bob revela informações valiosas antes de Trudy fornecer alguma prova de quem é ela;
2. fazer que o transmissor e o receptor utilizem chaves específicas para provar quem são, mesmo que isso signifique ter duas chaves compartilhadas, K_{AB} e K'_{AB};
3. fazer que o transmissor e o receptor extraiam seus desafios de conjuntos distintos. Por exemplo, o transmissor deve usar números pares e o receptor deve usar números ímpares.
4. Tornar o protocolo resistente a ataques que envolvam uma segunda sessão paralela, nos quais as informações obtidas em uma sessão são usadas em uma sessão diferente.

Figura 8.29 | Um protocolo de autenticação bidirecional reduzido.

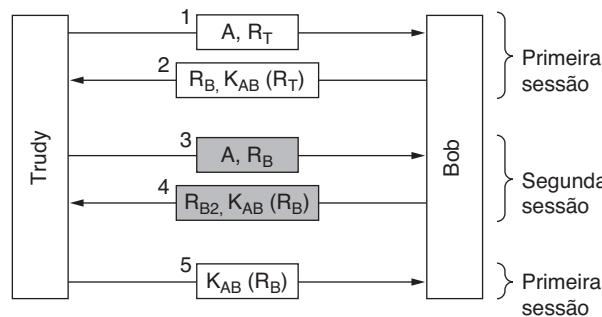



Figura 8.30 | O ataque por reflexão.

Se apenas uma dessas regras for violada, isso significa que o protocolo poderá ser violado com frequência. Nesse caso, todas as quatro regras foram violadas, com consequências desastrosas.

Agora, vamos examinar mais de perto a Figura 8.28. É possível garantir que esse protocolo não esteja sujeito a um ataque por reflexão? Bem, talvez. Essa é uma questão bastante sutil. Trudy foi capaz de violar nosso protocolo usando um ataque por reflexão, porque foi possível abrir uma segunda sessão com Bob e enganá-lo, respondendo a suas próprias perguntas. O que aconteceria se Alice fosse um computador de uso geral que também aceitasse várias sessões, em vez de ser uma pessoa diante de um computador? Vejamos o que Trudy poderia fazer nesse caso.

Para ver como funciona o ataque de Trudy, observe a Figura 8.31. Alice começa anunciando sua identidade na mensagem 1. Trudy intercepta essa mensagem e inicia sua própria sessão com a mensagem 2, afirmando ser Bob. Novamente sombreamos as mensagens da sessão 2. Alice responde a mensagem 2 com a mensagem 3, dizendo: 'Você afirma ser Bob? Então, prove'. Nesse momento Trudy não tem saída, porque não pode provar ser Bob.

O que Trudy faz agora? Ela volta para a primeira sessão, já que é sua vez de enviar um desafio, e transmite a R_A que obteve na mensagem 3. Alice gentilmente responde na mensagem 5 e, desse modo, fornece a Trudy as informações de que ela precisa para enviar a mensagem 6 na sessão 2. Nesse momento, Trudy está à vontade, porque conseguiu responder com sucesso ao desafio de Alice na sessão 2. Agora ela pode cancelar a sessão 1, transmitir qualquer número antigo para o restante da sessão 2, e terá uma sessão autenticada com Alice na sessão 2.

Figura 8.31 | Um ataque por reflexão no protocolo da Figura 8.28.

Contudo, Trudy é malvada e quer causar ainda mais danos. Em vez de enviar qualquer número antigo para concluir a sessão 2, ela espera até Alice enviar a mensagem 7, o desafio de Alice para a sessão 1. É claro que Trudy não sabe como responder, e portanto utiliza outra vez o ataque por reflexão, devolvendo R_{A2} como a mensagem 8. Alice codifica R_{A2} de maneira conveniente na mensagem 9. Agora, Trudy volta para a sessão 1 e envia a Alice, na mensagem 10, o número que ela deseja, cuidadosamente copiado do número que a própria Alice enviou na mensagem 9. Nesse momento, Trudy tem duas sessões completamente autenticadas com Alice.

Esse ataque tem um resultado um pouco diferente do ataque no protocolo de três mensagens, ilustrado na Figura 8.30. Dessa vez, Trudy tem duas conexões autenticadas com Alice. No exemplo anterior, ela tinha uma conexão autenticada com Bob. Mais uma vez, se aplicássemos todas as regras gerais de protocolos de autenticação discutidas anteriormente, esse ataque poderia ter sido interrompido. Uma descrição detalhada desses tipos de ataques e de como frustrá-los é apresentada em Bird et al. (1993), que também mostram como é possível construir, de forma sistemática, protocolos que comprovadamente são corretos. Porém, o mais simples desses protocolos é um pouco complicado; portanto, mostraremos agora uma classe de protocolo diferente, que também funciona.

O novo protocolo de autenticação é mostrado na Figura 8.32 (Bird et al., 1993). Ele emprega um HMAC do tipo que vimos quando estudamos o IPsec. Alice começa enviando a Bob um nonce R_A como mensagem 1. Bob responde selecionando seu próprio nonce, R_B, e devolvendo-o juntamente com um HMAC. O HMAC é formado com o objetivo de construir uma estrutura de dados que consiste no nonce de Alice, no nonce de Bob, em suas identidades e na chave secreta compartilhada, K_{AB}. Esses dados estruturados passam então por um hash no HMAC, por exemplo, usando SHA-1. Quando receber a mensagem 2, Alice terá R_A (que ela própria escolheu), R_B, que chegará sob a forma de texto simples, as duas identidades e a chave secreta K_{AB}, conhecida desde o início, e depois ela mesma poderá calcular o HMAC. Se este corresponder ao HMAC da mensagem, ela saberá que está se comunicando com Bob, porque Trudy não conhece K_{AB} e, desse modo, não terá como saber qual HMAC enviar. Alice responde a Bob com um HMAC contendo apenas os dois nonces.

Trudy pode subverter de algum modo esse protocolo? Não, porque ela não é capaz de forçar nenhuma das partes a codificar ou fazer o hash de um valor de sua escolha, como aconteceu na Figura 8.30 e na Figura 8.31. Ambos os HMACs incluem valores escolhidos pela parte transmissora, algo que Trudy não pode controlar.

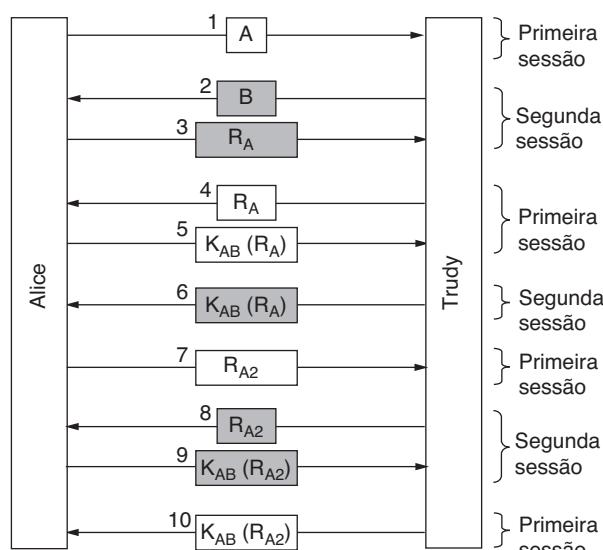
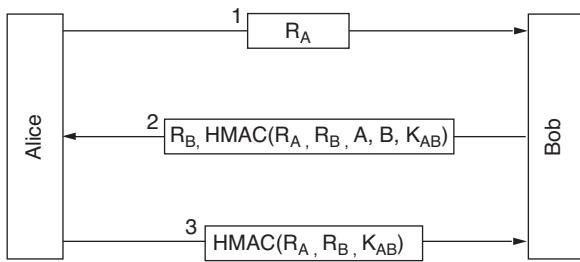



Figura 8.32 | Autenticação usando HMACs.

Utilizar HMACs não é o único meio de empregar essa ideia. Um esquema alternativo usado com frequência em vez de calcular o HMAC sobre uma série de itens é codificar os itens em sequência utilizando o encadeamento de blocos de cifras.

8.7.2 Como estabelecer chave compartilhada: a troca de chaves de Diffie-Hellman

Até agora, partimos do princípio de que Alice e Bob compartilham uma chave secreta. Suponha que isso não seja verdade (porque até agora não há nenhuma PKI universalmente aceita para assinar e distribuir certificados). Como eles podem estabelecer uma chave secreta? Uma possibilidade seria Alice telefonar para Bob e dar sua chave a ele, mas provavelmente ele começaria a conversa dizendo: 'Como posso saber que você é Alice e não Trudy?' Eles poderiam tentar se encontrar pessoalmente, com cada um levando passaporte, carteira de identidade e três cartões de crédito. No entanto, como são muito ocupados, talvez não consigam encontrar uma data conveniente para ambos durante meses. Felizmente, apesar de parecer incrível, há uma forma de pessoas que não se conhecem estabelecerem uma chave secreta em plena luz do dia, mesmo com Trudy registrando cuidadosamente cada mensagem.

O protocolo que permite o estabelecimento de uma chave secreta entre pessoas que não se conhecem é chamado **troca de chaves de Diffie-Hellman** (Diffie e Hellman, 1976) e funciona da forma descrita a seguir. Alice e Bob têm de concordar em relação a dois números grandes, n e g, onde n é um número primo, $(n - 1)/2$ também é um número primo e onde certas condições se aplicam a g. Esses números podem ser públicos; portanto, um deles só precisa selecionar n e g e informar ao outro abertamente. Agora, Alice escolhe um número grande x (digamos, de 1.024 bits) e o mantém em segredo. Da mesma forma, Bob seleciona um número grande secreto, y.

Alice inicia um protocolo de troca de chaves enviando a Bob uma mensagem contendo $(n, g, g^x \bmod n)$, como mostra a Figura 8.33. Bob responde enviando a Alice uma mensagem contendo $g^y \bmod n$. Agora, Alice eleva à x-ésima potência em módulo n o número que Bob lhe enviou, a fim de obter $(g^y \bmod n)^x \bmod n$. Bob executa uma operação semelhante para obter $(g^x \bmod n)^y \bmod n$. Pelas leis da aritmética modular, os dois cálculos geram $g^{xy} \bmod n$. Eis que, como por mágica, Alice e Bob de repente compartilham uma chave secreta, $g^{xy} \bmod n$.

É obvio que Trudy viu as duas mensagens. Com base na mensagem 1, ela conhece g e n. Se pudesse calcular x e y, Trudy poderia descobrir a chave secreta. O problema é que, com apenas $g^x \bmod n$, ela não consegue encontrar x. Não existem algoritmos práticos para o cálculo de logaritmos discretos cuja base é um número primo muito grande.

Para tornar o exemplo mostrado anteriormente mais concreto, utilizaremos os valores (completamente falsos): $n = 47$ e $g = 3$. Alice seleciona $x = 8$ e Bob seleciona $y = 10$, o que é mantido em segredo. A mensagem de Alice para Bob é (47, 3, 28), pois $3^8 \bmod 47$ é igual a 28. A mensagem de Bob para Alice é (17). Alice calcula $17^8 \bmod 47$, que é igual a 4. Bob calcula $28^{10} \bmod 47$, que é igual a 4. Alice e Bob determinaram de forma independente que agora a chave secreta é 4. Para descobrir a chave, Trudy tem de resolver a equação $3^x \bmod 47 = 28$, o que pode ser feito por pesquisa exaustiva de números pequenos como esse, mas não quando todos os números têm centenas de bits. Todos os algoritmos conhecidos até o momento demoram muito tempo para realizar esse cálculo, mesmo em supercomputadores paralelos e supervelozes.

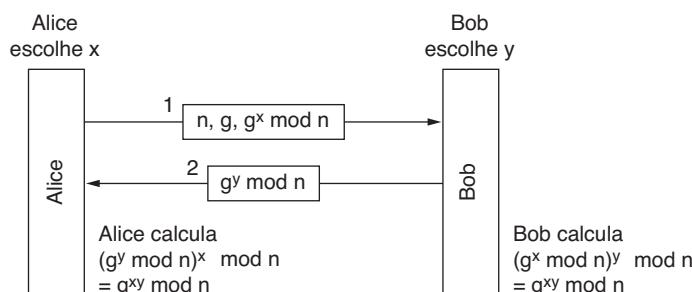



Figura 8.33 | A troca de chaves de Diffie-Hellman.

Apesar da elegância do algoritmo de Diffie-Hellman, há um problema: quando Bob obtém a tripla (47, 3, 28), como ele pode saber se ela veio de Alice e não de Trudy? Ele não tem mesmo como saber. Infelizmente, Trudy pode explorar esse fato para enganar Alice e Bob, como ilustra a Figura 8.34. Aqui, enquanto Alice e Bob escolhem *x* e *y*, respectivamente, Trudy escolhe seu próprio número aleatório, *z*. Alice envia a mensagem 1 para Bob. Trudy a intercepta e envia a mensagem 2 para Bob, usando *g* e *n* corretos (que, de qualquer forma, são públicos), mas com seu próprio *z* em vez de *x*. Ela também envia a mensagem 3 de volta para Alice. Mais tarde, Bob envia a mensagem 4 para Alice, que Trudy mais uma vez intercepta e guarda.

Agora, todos utilizam a aritmética modular. Alice calcula a chave secreta como g^{xz} mod *n*, e Trudy faz o mesmo (para mensagens enviadas a Alice). Bob calcula g^{yz} mod *n* e Trudy também (nas mensagens enviadas a Bob). Alice pensa que se comunica com Bob e, portanto, estabelece uma chave de sessão (com Trudy). Bob faz o mesmo. Todas as mensagens que Alice envia na sessão criptografada são capturadas, armazenadas e modificadas por Trudy para então (talvez) ser passadas a Bob. Da mesma forma, no outro sentido, Trudy vê tudo e pode modificar todas as mensagens como quiser, enquanto Alice e Bob têm a ilusão de que há um canal seguro entre os dois. Por isso, esse ataque é chamado **ataque do homem no meio**. Ele também é chamado **ataque da brigada de incêndio**, pois lembra um antigo corpo de bombeiros formado por voluntários que, enfileirados, passavam baldes d'água de mão em mão do caminhão até o incêndio.

8.7.3 Autenticação com o uso de um centro de distribuição de chaves

A ideia de compartilhar um segredo com uma pessoa estranha quase funcionou. Por outro lado, talvez não tenha valido a pena a tentativa (ataque da raposa e das uvas). Para conversar com *n* pessoas dessa forma, você precisará de *n* chaves. Para pessoas famosas, o gerenciamento de chaves poderia se tornar uma grande dor de cabeça, especialmente se cada chave tivesse de ser guardada em um cartão plástico com chip embutido.

Outra estratégia é introduzir um centro de distribuição de chaves (KDC — Key Distribution Center) confiável. Nesse modelo, cada usuário tem uma única chave compartilhada com o KDC. Agora o gerenciamento de sessão e de autenticação passa pelo KDC. O protocolo de autenticação para o KDC mais simples envolve duas partes e um KDC confiável, e é descrito na Figura 8.35.

A ideia por trás desse protocolo é simples: Alice escolhe uma chave de sessão K_S e informa ao KDC que deseja se comunicar com Bob usando K_S. Essa mensagem é criptografada com a chave secreta que Alice compartilha (apenas) com o KDC, K_A. O KDC a descriptografa e extrai a identidade de Bob e a chave de sessão. Em seguida, cria uma nova mensagem contendo a identidade de Alice e a chave de sessão, depois envia essa mensagem a Bob. A criptografia é feita com K_B, a chave secreta que Bob compartilha com o KDC. Quando descriptografa a mensagem, Bob fica sabendo que Alice quer se comunicar com ele e qual chave ela desejar usar.

Aqui, a autenticação ocorre sem maiores problemas. O KDC sabe que a mensagem 1 deve vir de Alice, pois ninguém mais teria sido capaz de criptografá-la com a chave secreta de Alice. Da mesma forma, Bob sabe que a mensagem 2 deve ter vindo do KDC, em quem ele confia, pois ninguém mais conhece sua chave secreta.

Infelizmente, esse protocolo tem uma falha muito séria. Trudy precisa de dinheiro; portanto, ela descobre alguns serviços legítimos que pode prestar a Alice, faz uma oferta interessante e consegue o trabalho. Depois de fazer o serviço, Trudy educadamente solicita que Alice pague por transferência bancária. Alice estabelece uma chave de sessão com o funcionário do banco, Bob. Em seguida, envia a Bob uma mensagem solicitando que o dinheiro seja transferido para a conta de Trudy.

Nesse ínterim, Trudy volta a bisbilhotar a rede. Ela copia tanto a mensagem 2 da Figura 8.35 quanto a solicitação de transferência de dinheiro enviada depois da mensagem.

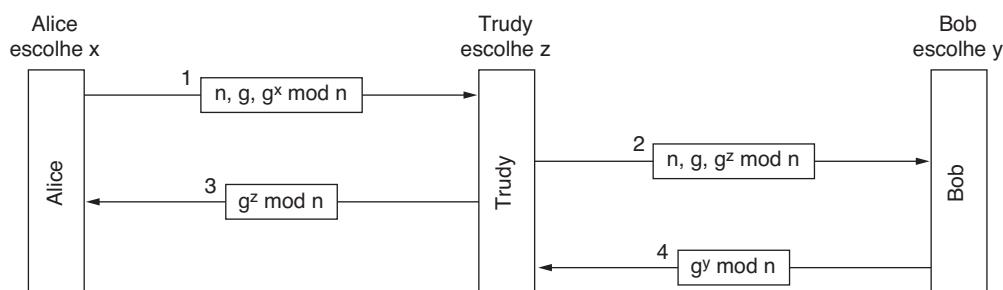



Figura 8.34 | O ataque do homem no meio.

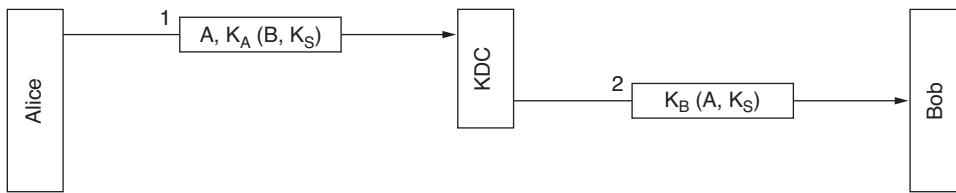



Figura 8.35 | Uma primeira tentativa de protocolo de autenticação usando um KDC.

Mais tarde, ela responde a ambas as mensagens de Bob. Ele as recebe e pensa: 'Alice deve ter contratado Trudy outra vez. Percebe-se que o trabalho dela é muito bom'. Em seguida, Bob transfere uma quantia igual em dinheiro da conta de Alice para a conta de Trudy. Algum tempo depois do 50º par de mensagens, Bob vai até Trudy e oferece um bom empréstimo para que ela possa expandir seus negócios, que obviamente vão muito bem. Esse problema é chamado **ataque por replay**.

Existem várias soluções para o ataque por replay. A primeira é incluir um registro de tempo em cada mensagem. Então, se alguém receber uma mensagem obsoleta, ela poderá ser descartada. O problema é que os clocks nunca estão sincronizados com exatidão na rede; portanto, deve haver um período durante o qual esse registro de tempo será válido. Trudy pode repetir a mensagem durante esse período e se livrar dela.

A segunda solução é colocar um nonce em cada mensagem. Nesse caso, cada parte terá de se lembrar de todos os nonces anteriores e rejeitar as mensagens que contenham algum já utilizado. No entanto, os nonces têm de ser memorizados para sempre, por receio de que Trudy tente repetir uma mensagem de cinco anos. Além disso, se alguma máquina apresentar falha e perder sua lista de nonces, ela estará vulnerável a um ataque por replay. Os registros de tempo e os nonces podem ser combinados para limitar o tempo durante o qual estes têm de ser memorizados, mas é óbvio que o protocolo ficará muito mais complicado.

Um enfoque mais sofisticado para a autenticação mútua é usar um protocolo de desafio-resposta que funcione em diversas direções. Um exemplo bastante conhecido é o protocolo de **autenticação de Needham-Schroeder** (Needham e Schroeder, 1978), do qual uma das variantes é mostrada na Figura 8.36.

O protocolo começa com Alice informando ao KDC que deseja se comunicar com Bob. Essa mensagem contém um número grande aleatório, R_A, que é usado como nonce. O KDC retorna a mensagem 2 contendo o número aleatório de Alice, uma chave de sessão e um bilhete que ela pode enviar a Bob. O objetivo do número aleatório, R_A, é garantir a Alice que a mensagem 2 é nova e não um replay. A identidade de Bob também é enviada, caso Trudy pense na possibilidade de substituir B na mensagem 1 por sua própria identidade, para que o KDC codifique o bilhete no fim da mensagem 2 com K_T em vez de K_B. O bilhete codificado com K_B é incluído na mensagem criptografada para impedir que Trudy o substitua por algo diferente quando ele retornar a Alice.

Agora Alice envia o bilhete a Bob, junto com um novo número aleatório, R_{A2}, criptografado com a chave de sessão, K_S. Na mensagem 4, Bob envia K_S (R_{A2} − 1), a fim de provar a Alice que ela está se comunicando com o verdadeiro Bob. Retornar K_S (R_{A2}) não teria funcionado, pois Trudy poderia ter acabado de roubar essa chave na mensagem 3.

Depois de receber a mensagem 4, Alice estará convencida de que está se comunicando com Bob e de que nenhum replay poderia ter sido usado até então. Afinal, ela acabou de gerar R_{A2} alguns milissegundos antes. O objetivo

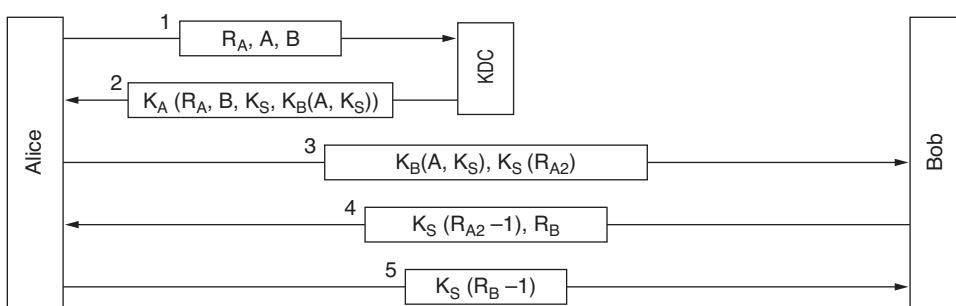



Figura 8.36 | O protocolo de autenticação Needham-Schroeder.

da mensagem 5 é convencer Bob de que ele está se comunicando realmente com Alice, e que nenhum replay está sendo usado aqui. Ao fazer com que cada parte gere um desafio e responda a outro, a possibilidade de qualquer tipo de ataque por replay é eliminada.

Apesar de parecer bastante sólido, esse protocolo tem uma pequena falha. Se Trudy conseguir obter uma antiga chave de sessão em texto simples, poderá iniciar uma nova sessão com Bob repetindo a mensagem 3 correspondente à chave comprometida e convencê-lo de que é Alice (Denning e Sacco, 1981). Dessa vez, ela poderá desfalcar a conta bancária de Alice sem precisar prestar nenhum serviço honesto.

Mais tarde, Needham e Schroeder (1987) publicaram um protocolo que corrige esse problema. No mesmo exemplar do mesmo periódico, Otway e Rees (1987) também publicaram um protocolo que resolve o problema de uma forma mais simples. A Figura 8.37 mostra o protocolo de Otway-Rees ligeiramente modificado.

No protocolo de Otway-Rees, Alice começa gerando um par de números aleatórios, *R*, que será usado como um identificador comum, R_A, que Alice utilizará para desafiar Bob. Quando receber essa mensagem, Bob criará uma nova com a parte criptografada da de Alice e mais uma semelhante de sua própria autoria. Ambas as partes criptografadas com K_A e K_B identificam Alice e Bob, e contêm o identificador comum e um desafio.

O KDC verifica se o *R* de ambas as partes é igual. Talvez não seja, porque Trudy adulterou *R* na mensagem 1 ou substituiu parte da mensagem 2. Se os dois números *R* coincidirem, o KDC considerará válida a mensagem de solicitação de Bob. Em seguida, o KDC vai gerar uma chave de sessão criptografada duas vezes, uma para Alice e outra para Bob. Cada mensagem conterá o número aleatório do receptor, como prova de que o KDC, e não Trudy, gerou a mensagem. Nesse momento, tanto Alice quanto Bob têm a mesma chave de sessão e podem começar a comunicação. Na primeira vez que eles trocarem mensagens de dados, cada um poderá ver que o outro tem uma cópia idêntica de K_S, e assim a autenticação é concluída.

8.7.4 Autenticação com a utilização do Kerberos

Um protocolo de autenticação usado em muitos sistemas reais (inclusive no Windows 2000 e versões posteriores) é o **Kerberos**, que se baseia em uma variante do protocolo de Needham-Schroeder. Seu nome se deve ao cão de várias cabeças da mitologia grega que guardava a entrada do Hades (provavelmente para manter as pessoas indesejáveis à distância). O Kerberos foi projetado no MIT para permitir que os usuários de estações de trabalho tivessem acesso a recursos da rede de uma forma segura. Sua grande diferença em relação ao protocolo de Needham-Schroeder é a suposição de que todos os clocks estão muito bem sincronizados. O protocolo passou por várias iterações. A V5 é a versão mais usada na indústria, e está definida na RFC 4120. A anterior, V4, por fim, foi retirada após sérias falhas (Yu et al., 2004). A V5 melhora a V4 com muitas mudanças pequenas no protocolo e alguns recursos melhorados, como o fato de não contar mais com o DES, agora desatualizado. Para obter mais informações, consulte Neuman e Ts'o (1994).

O Kerberos envolve três servidores além de Alice (uma estação de trabalho cliente):

1. o Authentication Server (AS): verifica a identidade dos usuários durante o processo de login;
2. o Ticket-Granting Server (TGS): emite 'bilhetes de comprovação de identidade';
3. o servidor Bob: na verdade, faz o trabalho que Alice deseja ver pronto.

O AS é semelhante a um KDC, porque compartilha uma senha secreta com todos os usuários. O trabalho do TGS é emitir bilhetes que possam convencer os servidores reais de que o portador de um bilhete TGS realmente é quem afirma ser.

Para iniciar uma sessão, Alice utiliza uma estação de trabalho pública qualquer e digita seu nome. A estação de trabalho envia seu nome e o do TGS ao AS em texto simples, como mostra a mensagem 1 da Figura 8.38. O retorno é uma chave de sessão e um bilhete, K_{TGS} (A, K_S, t), destinado ao TGS. A chave de sessão é codificada com a chave secreta de Alice, de modo que apenas Alice possa decodifi-

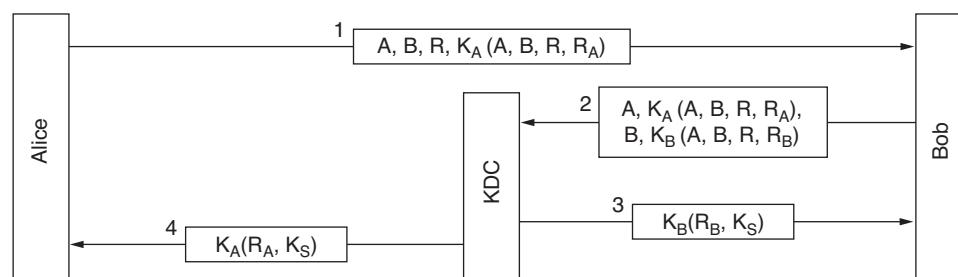



Figura 8.37 | O protocolo de autenticação de Otway-Rees (ligeiramente simplificado).

cá-la. Somente quando a mensagem 2 chega, a estação de trabalho pede a senha de Alice — não antes. Em seguida, a senha é usada para gerar K_A, a fim de descriptografar a mensagem 2 e obter a chave de sessão.

Nesse momento, a estação de trabalho substitui a senha de Alice, para garantir que a senha só estará na estação durante alguns milissegundos, no máximo. Se Trudy tentar estabelecer um login como Alice, a senha que ela digitar estará errada e a estação de trabalho detectará o problema, porque o trecho-padrão da mensagem 2 estará incorreto.

Depois de estabelecer o login, Alice pode informar à estação de trabalho que deseja contatar Bob, o servidor de arquivos. Em seguida, a estação de trabalho envia a mensagem 3 ao TGS solicitando um bilhete para usar com Bob. O principal elemento nessa solicitação é K_{TGS} (A, K_S, t), criptografado com a chave secreta de TGS e usado como prova de que o transmissor realmente é Alice. O TGS responde criando uma chave de sessão, K_{AB}, para que Alice a utilize com Bob. Duas versões dessa chave são retornadas. A primeira é criptografada apenas com K_S, para que Alice possa ler a mensagem. A segunda, com a chave de Bob, K_B, de forma que ele também possa ler a mensagem.

Trudy pode copiar a mensagem 3 e tentar usá-la mais uma vez, mas será frustrada pelo registro de tempo criptografada, *t*, enviada junto. Trudy não pode substituir o registro de tempo por outro mais recente, porque não conhece K_S, a chave de sessão que Alice utiliza para se comunicar com o TGS. Mesmo que Trudy repita a mensagem 3 rapidamente, tudo o que obterá será outra cópia da mensagem 4 — que não pôde descriptografar antes e que também não poderá descriptografar no momento.

Então, Alice envia K_{AB} a Bob, para estabelecer uma sessão com ele. Essa troca também recebe um registro de tempo. A resposta é a prova de que Alice está realmente se comunicando com Bob, e não com Trudy.

Depois de uma série de trocas, Alice pode se comunicar com Bob sob a proteção de K_{AB}. Se mais tarde ela chegar à conclusão de que precisa se comunicar com outro servidor, Carol, Alice simplesmente repetirá a mensagem 3 para o TGS, apenas especificando *C* em vez de *B*. O TGS responderá prontamente com um bilhete criptografado com K_C, que Alice poderá enviar a Carol e que Carol aceitará como prova de que Alice o enviou.

O objetivo de todo esse trabalho é que agora Alice pode acessar servidores instalados por toda a rede de forma segura, e sua senha nunca terá de percorrer a rede. Na verdade, a senha só precisaria permanecer na estação de trabalho da própria Alice durante alguns milissegundos. No entanto, observe que cada servidor faz sua própria autorização. Quando Alice apresenta seu bilhete a Bob, isso prova a ele quem o enviou. Na verdade, a decisão sobre o que Alice está autorizada a fazer cabe a Bob.

Como os projetistas do Kerberos não esperavam que o mundo inteiro confiasse em um único servidor de autenticação, reservaram espaço para o uso de diversos **realms** (domínios), cada um com seu próprio AS e TGS. Para obter um bilhete referente a um servidor de um domínio distante, Alice solicitaria a seu próprio TGS um bilhete aceito pelo TGS do domínio distante. Se o TGS distante tiver sido registrado com o local (exatamente como fazem os servidores locais), este dará a Alice um bilhete válido no TGS distante. Depois disso, ela poderá usá-lo para executar ações como obter bilhetes para os servidores desse domínio. No entanto, observe que, para partes pertencentes a dois domínios interagirem, cada uma deverá confiar no TGS da outra. Caso contrário, não haverá interação.

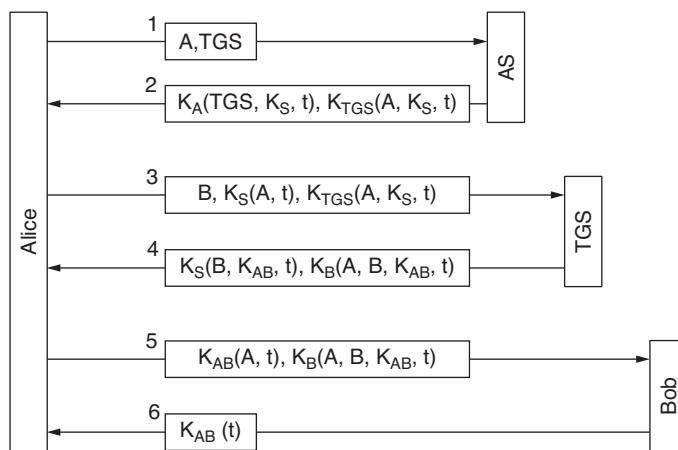



Figura 8.38 | A operação do Kerberos V5.

8.7.5 Autenticação com a criptografia de chave pública

Também é possível fazer uma autenticação mútua com o uso da criptografia de chave pública. Para começar, Alice precisa da chave pública de Bob. Se existir uma PKI com o servidor de diretórios que entregue certificados para chaves públicas, Alice poderá solicitar o de Bob, como mostra a Figura 8.39, na mensagem 1. A resposta, na mensagem 2, é um certificado X.509 que contém a chave pública de Bob. Quando verifica que a assinatura está correta, Alice envia a Bob uma mensagem contendo sua identidade e um nonce.

Quando recebe essa mensagem, Bob não sabe com certeza se ela veio de Alice ou de Trudy, mas continua e pede ao servidor de diretórios a chave pública de Alice (mensagem 4) e logo a recebe (mensagem 5). Em seguida, envia a Alice uma mensagem contendo o R_A de Alice, seu próprio nonce, R_B, e uma chave de sessão sugerida, K_S.

Quando recebe a mensagem 6, Alice a descriptografa usando sua própria chave privada. Ela vê R_A na mensagem e fica feliz. A mensagem deve ter vindo de Bob, pois Trudy não tem como determinar R_A. Além disso, a mensagem deve ser nova, e não um replay, pois ela acabou de enviar R_A a Bob. Alice concorda com a sessão retornando a mensagem 7. Quando vê R_B criptografada com a chave de sessão que acabou de gerar, Bob fica sabendo que Alice recebeu a mensagem 6 e confirmou R_A. Agora Bob está feliz.

O que Trudy pode fazer para subverter esse protocolo? Ela pode falsificar a mensagem 3 e enganar Bob fazendo-o pensar que ela é Alice, mas Alice verá um R_A que não enviou e não prosseguirá com a transmissão. Trudy não poderá forjar a mensagem 7 de volta para Bob, pois não conhece os valores de R_B e de K_S e não pode determiná-los sem a chave privada de Alice. Ela está sem sorte.

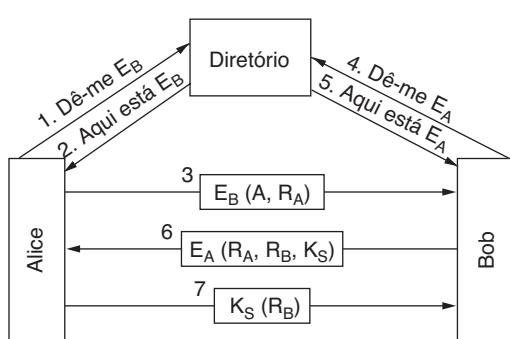



Figura 8.39 | Autenticação mútua com a utilização da criptografia de chave pública.

8.8 Segurança de correio eletrônico

Quando uma mensagem de correio eletrônico é enviada entre dois sites distantes, geralmente ela transita por dezenas de máquinas até chegar a seu destino. Qualquer uma dessas máquinas pode ler e armazenar a mensagem para usá-la posteriormente. Na prática, não há privacidade, apesar de muita gente achar o contrário. Todavia, muita gente gostaria de enviar mensagens de correio eletrônico para que fossem lidas pelo destinatário pretendido e por ninguém mais (nem seu chefe, nem o governo). Esse desejo estimulou muitas pessoas e grupos a aplicar os princípios da criptografia que estudamos anteriormente para produzir mensagens seguras. Nas seções a seguir, estudaremos um sistema de correio eletrônico seguro e bastante utilizado, o PGP, e depois mencionaremos brevemente outro sistema, o S/MIME. Para obter mais informações sobre correio eletrônico seguro, consulte Kaufman et al. (2002) e Schneier (1995).

8.8.1 PGP — Pretty Good Privacy

Nosso primeiro exemplo, o **PGP** (**Pretty Good Privacy**), foi criado por uma única pessoa, Phil Zimmermann (1995a, 1995b). Zimmermann é um defensor da privacidade cujo lema é: 'Se a privacidade for ilegal, somente os ilegais terão privacidade'. Lançado em 1991, o PGP é um pacote completo para segurança de mensagens de correio eletrônico que fornece privacidade, autenticação, assinaturas digitais e compactação, tudo de uma forma fácil de usar. Além disso, o pacote completo, incluindo o código-fonte, é distribuído de graça via Internet. Graças à qualidade, ao preço (zero) e à disponibilidade em plataformas UNIX, Linux, Windows e Mac OS, é um sistema bastante utilizado hoje.

O PGP codifica dados usando uma cifra de bloco chamada **IDEA** (**International Data Encryption Algorithm**), que utiliza chaves de 128 bits. Ele foi criado na Suíça em uma época na qual o DES era considerado decadente e o AES ainda não tinha surgido. Em termos conceituais, o IDEA é semelhante ao DES e ao AES: ele embaralha os bits em uma série de rodadas, mas os detalhes das funções executoras são diferentes dos de DES e AES. O gerenciamento de chaves utiliza o RSA e a integridade de dados utiliza o MD5, tópicos que já discutimos.

O PGP também esteve envolvido em controvérsias desde o início (Levy, 1993). Como Zimmermann não fez nada para impedir que outras pessoas colocassem o PGP na Internet, onde gente de todo o mundo poderia obtê-lo, o governo dos Estados Unidos afirmou que Zimmermann violou leis norte-americanas que proibiam a exportação de munições. A investigação durou cinco anos, mas foi abandonada, provavelmente por dois motivos. Primeiro, Zimmermann não colocou o PGP na Internet, e assim seu advogado afirmou que *ele* nunca exportou nada (e, na época, havia dúvidas sobre se criar um site constituiria uma forma de exportação).

Segundo, o governo percebeu mais tarde que vencer uma disputa judicial significava convencer um júri de que um site contendo um programa de privacidade passível de ser transferido por download era uma infração sujeita às penas da lei contra tráfico de armas — que proibia a exportação de materiais de guerra como tanques, submarinos, aeronaves militares e armas nucleares. Vários anos de publicidade negativa provavelmente também não ajudariam muito.

A propósito, as regras de exportação são bizarras, para dizer o mínimo. O governo considerava a colocação de código em um site um ato de exportação ilegal e processou Zimmermann durante cinco anos. Por outro lado, quando alguém publicava o código-fonte completo do PGP em linguagem C sob a forma de um livro (em uma fonte de tamanho grande com um checksum em cada página, para facilitar a digitalização) e depois exportava o livro, isso era legal, porque os livros não são classificados como munições. A espada é mais poderosa que a caneta, pelo menos para o Tio Sam.

Outro problema que o PGP enfrentou envolvia a violação de patentes. A empresa que detinha a patente do RSA, denominada RSA Security, Inc., alegou que o uso que o PGP fazia do algoritmo RSA infringia sua patente, mas esse problema foi contornado nas versões seguintes, a partir da versão 2.6. Além disso, o PGP usa outro algoritmo de criptografia patenteado, o IDEA, o que causou alguns problemas no início.

Tendo em vista que o PGP tem código-fonte aberto, muitas pessoas e grupos o modificaram e produziram várias versões. Algumas delas foram projetadas para contornar as leis de munições; outras se concentraram em evitar o uso de algoritmos patenteados; e ainda outras queriam transformá-lo em um produto comercial com código-fonte fechado. Embora as leis de munições tenham sido ligeiramente atenuadas (do contrário, produtos que usassem o AES não poderiam ter sido exportados pelos Estados Unidos) e a patente do RSA tenha expirado em setembro de 2000, o legado de todos esses problemas foi a existência de várias versões incompatíveis do PGP, identificadas por diversos nomes. A descrição a seguir se concentra no PGP clássico, a versão mais antiga e mais simples. Outra versão popular, o Open PGP, é descrita na RFC 2440. Ainda outra é o Privacy Guard do GNU.

O PGP utilizou intencionalmente algoritmos criptográficos que já existiam, em vez de criar novos. Ele se baseia em algoritmos que passaram por intensas revisões e não foram projetados ou influenciados por nenhuma agência governamental que tentasse enfraquecê-los. Para pessoas que tendem a não acreditar no governo, essa característica representa uma excelente opção.

O PGP aceita compactação de textos, sigilo e assinaturas digitais, e também oferece amplos recursos de gerenciamento de chaves, mas, estranhamente, não oferece recursos de correio eletrônico. Ele é mais parecido com um pré-processador que recebe texto simples como entrada e produz texto cifrado assinado em base64 como saída. Essa saída pode então ser enviada por correio eletrônico. Algumas implementações do PGP chamam um agente do usuário na etapa final para enviar de fato a mensagem.

Para ver como o PGP funciona, considere o exemplo da Figura 8.40. Alice deseja enviar uma mensagem em texto simples assinada, *P*, para Bob de forma segura. Tanto Alice quanto Bob têm chaves RSA privadas (D_X) e públicas (E_X). Vamos supor que cada um conheça a chave pública do outro; examinaremos em breve o gerenciamento de chaves do PGP.

Alice começa invocando o programa PGP em seu computador. Primeiro, o PGP submete sua mensagem *P* a um processo de hash, utilizando o MD5; em seguida, criptografa o resultado com sua chave privada RSA, D_A. Quando receber a mensagem, Bob poderá descriptografar o hash com

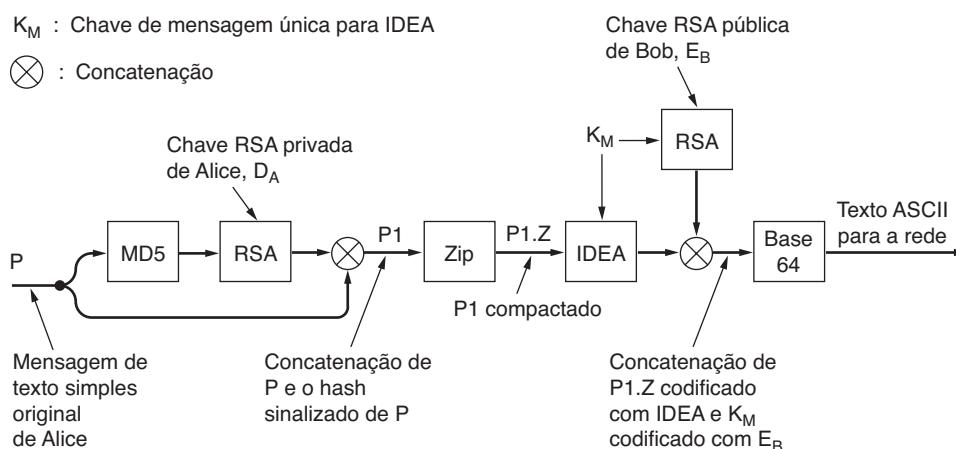



Figura 8.40 | O PGP em operação para enviar uma mensagem.

a chave pública de Alice e confirmar que o hash está correto. Mesmo que alguma outra pessoa (por exemplo, Trudy) pudesse adquirir o hash nesse estágio e descriptografá-lo com a chave pública de Alice, a robustez do MD5 garante que seria inviável em termos computacionais produzir outra mensagem com o mesmo hash MD5.

O hash criptografado e a mensagem original são concatenados em uma única mensagem *P1* e compactados com o programa ZIP, que emprega o algoritmo de Ziv-Lempel (Ziv e Lempel, 1977). Chame a saída dessa etapa de P1.Z.

Em seguida, o PGP solicita a Alice que informe dados aleatoriamente. O conteúdo e a velocidade de digitação são usados para gerar uma chave IDEA de 128 bits, K_M (denominada chave de sessão na literatura sobre o PGP; no entanto, essa denominação não é adequada, pois não há sessão). Em seguida, K_M é usada para criptografar *P1.Z* com o IDEA no modo de feedback de cifra. Além disso, K_M é criptografada com a chave pública de Bob, E_B. Em seguida, esses dois componentes são concatenados e convertidos para base64, como discutimos na seção sobre o MIME no Capítulo 7. A mensagem resultante contém apenas letras, dígitos e os símbolos +, / e =, o que significa que ela pode ser incluída em um corpo RFC 822 e chegar intacta a seu destino.

Ao receber a mensagem, Bob reverte a codificação base64 e decodifica a chave IDEA utilizando sua chave privada RSA. Ao usar essa chave, ele decodifica a mensagem para obter *P1.Z*. Após a descompactação, Bob separa o texto simples do hash criptografado e descriptografa o hash utilizando a chave pública de Alice. Se o hash de texto simples coincidir com seu próprio cálculo MD5, ele saberá que *P* é a mensagem correta e que ela veio de Alice.

Vale a pena observar que o RSA só é usado em duas situações: para criptografar o hash MD5 de 128 bits e para criptografar a chave IDEA de 128 bits. Apesar de o RSA ser lento, ele só precisa criptografar 256 bits, e não um grande volume de dados. Além disso, todos os 256 bits de texto simples são excessivamente aleatórios; portanto, Trudy terá muito trabalho para descobrir se uma suposta chave está correta. O trabalho de criptografia é feito pelo IDEA, que é várias ordens de grandeza mais rápido que o RSA. Portanto, o PGP oferece segurança, compactação e assinatura digital de uma forma muito mais eficiente do que o esquema ilustrado na Figura 8.16.

O PGP aceita quatro tamanhos de chaves RSA. Cabe ao usuário selecionar o mais apropriado. Os tamanhos são:

1. casual (384 bits): pode ser decifrado com facilidade atualmente.
2. comercial (512 bits): pode ser decifrado por empresas de informática.
3. militar (1.024 bits): ninguém no planeta consegue decifrar.
4. alienígena (2.048 bits): não pode ser decifrado por ninguém, nem de outros planetas.

Como o RSA só é usado para efetuar dois pequenos cálculos, todos deveriam usar chaves fortes alienígenas o tempo todo.

O formato de uma mensagem PGP clássica é mostrado na Figura 8.41. Diversos outros formatos também estão sendo usados. A mensagem tem três partes — a chave IDEA, a assinatura e a mensagem. A parte referente à chave contém não só a chave, mas também um identificador de chave, pois os usuários podem ter várias chaves públicas.

A parte referente à assinatura contém um cabeçalho, que não nos interessará aqui. O cabeçalho é seguido por um registro de tempo, pelo identificador da chave pública do transmissor — que pode ser usada para descriptografar o hash de assinatura — , algum tipo de informação que identifique os algoritmos utilizados (para permitir que o MD6 e o RSA2 sejam usados quando forem criados) e pelo próprio hash criptografado.

A parte referente à mensagem também contém um cabeçalho, o nome-padrão do arquivo a ser usado se o receptor gravar o arquivo no disco, o registro de tempo de criação da mensagem e, por fim, a própria mensagem.

No PGP, o gerenciamento de chaves recebeu muita atenção por ser o calcanhar de aquiles de todos os sistemas de segurança. O gerenciamento de chaves funciona da seguinte forma. Cada usuário mantém duas estruturas de da-

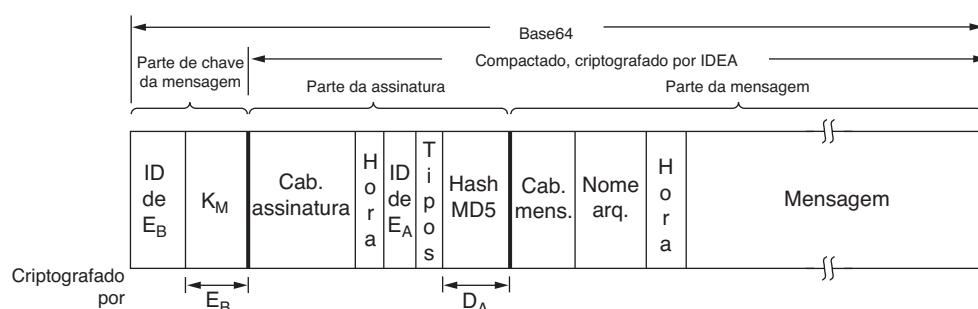



Figura 8.41 | Uma mensagem PGP.

dos localmente: um anel de chaves privadas e um anel de chaves públicas. O **anel de chaves privadas** contém um ou mais pares de chave pública/privada. A razão para aceitar vários pares por usuário é permitir que estes alterem suas chaves públicas periodicamente ou quando uma delas for considerada comprometida, sem invalidar as mensagens que estiverem sendo preparadas ou em trânsito. Cada par tem um identificador associado, para que o remetente da mensagem informe ao destinatário qual chave pública foi utilizada para criptografá-la. Os identificadores de mensagem consistem nos 64 bits de baixa ordem da chave pública. Os usuários são responsáveis por evitar conflitos em seus identificadores de chave pública. As chaves privadas armazenadas em disco são criptografadas com o uso de uma senha especial (arbitrariamente longa) para protegê-las contra ataques sorrateiros.

O **anel de chaves públicas** contém as chaves públicas correspondentes do usuário. Elas são necessárias para criptografar as chaves associadas a cada mensagem. Cada entrada do anel contém não só a chave pública, mas também seu identificador de 64 bits e uma indicação de até que ponto o usuário confia na chave.

O problema que está sendo resolvido é explicado a seguir. Vamos supor que as chaves públicas sejam mantidas em sistemas de BBS. Uma forma de Trudy ler a correspondência secreta de Bob é atacar o BBS e substituir a chave pública de Bob por outra de sua escolha. Quando Alice obtiver a chave que supostamente pertence a Bob, Trudy poderá montar um ataque do homem no meio (man-in-the-middle, bucket brigade attack) contra Bob.

Para impedir tais ataques, ou pelo menos minimizar suas consequências, Alice precisa saber até que ponto pode confiar no item ‘chave de Bob’ em seu anel de chaves públicas. Se souber que Bob entregou pessoalmente um disquete contendo a chave, ela poderá definir o valor de confiança como o mais alto. Essa é uma abordagem descentralizada e controlada pelo usuário para o gerenciamento de chaves públicas, o que distingue o PGP dos esquemas centralizados de PKI.

No entanto, na prática, frequentemente as pessoas recebem chaves públicas consultando um servidor de chaves confiável. Por essa razão, depois da padronização do X.509, o PGP também passou a admitir esses certificados, bem como o mecanismo tradicional de anel de chaves públicas do PGP. Todas as versões atuais do PGP têm suporte ao X.509.

8.8.2 S/MIME

O próximo empreendimento da IETF relacionado à segurança do correio eletrônico foi denominado **S/MIME (Secure/MIME)**, e é descrito nas RFCs 2632 a 2643. Ele oferece autenticação, integridade de dados, sigilo e não repúdio. É bastante flexível, pois admite uma variedade de algoritmos criptográficos. Considerando-se o nome, não surpreende que o S/MIME se integre bem ao MIME, permitindo que todos os tipos de mensagens sejam protegidos. Foi definida uma grande variedade de novos cabeçalhos MIME, por exemplo, para conter assinaturas digitais.

O S/MIME não tem uma estrutura rígida de certificados começando em uma única raiz, o que tem sido um dos problemas políticos que arruinaram um sistema mais antigo, chamado PEM (Privacy Enhanced Mail). Em vez disso, os usuários podem ter várias âncoras de confiança. Desde que a origem de um certificado possa ser acompanhada até alguma âncora de confiança em que o usuário acredite, ele é considerado válido. O S/MIME utiliza os algoritmos e protocolos-padrão que examinamos até agora; portanto, não o discutiremos mais aqui. Para ver os detalhes, consulte as RFCs.

8.9 Segurança da Web

Acabamos de estudar duas áreas importantes em que a segurança é necessária: comunicações e correio eletrônico. Considere-as entrada e aperitivo. Agora, vamos ao prato principal: a segurança da Web. O lugar onde encontramos a maioria dos intrusos, espionando e fazendo seu trabalho sujo é a Web. Nas próximas seções examinaremos alguns problemas e questões relacionados à segurança da Web.

Grosso modo, a segurança da Web pode ser dividida em três partes. Primeiro: como os objetos e os recursos são nomeados com segurança? Segundo: como é possível estabelecer conexões seguras e autenticadas? Terceiro: o que acontece quando um site envia a um cliente um fragmento de código executável? Depois de estudarmos algumas ameaças, vamos examinar todas essas questões.

8.9.1 Ameaças

Quase toda semana, lemos nos jornais notícias sobre problemas de segurança de sites. A situação é realmente bastante séria. Vamos examinar alguns exemplos do que já aconteceu. Primeiro, a home page de inúmeras organizações é atacada e substituída por uma nova home page escolhida pelos crackers. (A imprensa popular chama as pessoas que invadem computadores de ‘hackers’, mas muitos programadores reservam esse termo para os ótimos programadores. Preferimos chamar esses invasores de ‘crackers’.) Entre os sites invadidos incluem-se Yahoo, o Exército dos Estados Unidos, a CIA, a NASA e o *New York Times*. Na maioria dos casos, os crackers simplesmente colocavam algum texto engraçado, e os sites eram arrumados dentro de algumas horas.

Agora, vamos observar alguns casos muito mais sérios. Diversos sites foram derrubados por ataques de negação de serviço, nos quais o cracker inunda o site com tráfego, tornando-o incapaz de responder a consultas legítimas. Com frequência, o ataque é montado a partir de um grande número de máquinas que o cracker já invadiu (ataques DDoS). Esses ataques são tão comuns que já não geram mais notícias, mas podem custar ao site atacado milhares de dólares em negócios perdidos.

Em 1999, um cracker sueco invadiu o site Hotmail da Microsoft e criou um site-espelho que permitia a qualquer pessoa digitar o nome de um usuário do Hotmail e depois ler todo o correio eletrônico atual e arquivado da pessoa.

Em outro caso, um cracker russo de 19 anos chamado Maxim invadiu um site de comércio eletrônico e roubou 300 mil números de cartão de crédito. Em seguida, abordou os proprietários do site e informou que, se não recebesse 100 mil dólares, iria postar todos os números de cartões de crédito na Internet. Eles não cederam à chantagem, e o cracker realmente publicou os números dos cartões de crédito causando muitos danos a muitas vítimas inocentes.

Em um cenário diferente, um aluno de 23 anos na Califórnia enviou por correio eletrônico um comunicado a uma agência de notícias, afirmando falsamente que a Emulex Corporation iria anunciar um grande prejuízo trimestral e que o CEO da empresa renunciaria imediatamente. Dentro de poucas horas, as ações caíram 60%, fazendo os acionistas perder mais de 2 bilhões de dólares. O atacante ganhou 250.000 dólares vendendo suas ações pouco antes de enviar o anúncio. Embora esse evento não represente a invasão de um site, é claro que a inserção de um anúncio desse tipo na home page de qualquer grande corporação teria um efeito semelhante.

Poderíamos (infelizmente) continuar com esse assunto por muitas páginas, mas agora devemos examinar algumas questões técnicas relacionadas à segurança da Web. Para obter mais informações sobre problemas de segurança de todos os tipos, consulte Anderson (2008a); Stuttard e Pinto (2007); e Schneier (2004). A pesquisa na Internet também resultará na apresentação de um grande número de casos específicos.

8.9.2 Nomenclatura segura

Vamos iniciar com algo bastante básico: Alice quer visitar o site de Bob. Ela digita o URL de Bob em seu navegador e, em alguns segundos, surge uma página Web. Porém, será a página de Bob? Talvez sim e talvez não. Trudy poderia colocar em prática mais uma vez seus velhos truques. Por exemplo, ela poderia interceptar todos os pacotes enviados por Alice e examiná-los. Quando capturar uma solicitação *GET* do HTTP endereçada ao site de Bob, ela mesma poderá ir até o site de Bob para obter a página, modificá-la como desejar e retornar a Alice a página falsa. Alice nem ficaria sabendo. Pior ainda, Trudy poderia diminuir os preços da loja eletrônica de Bob para tornar suas mercadorias muito atraentes, fazendo Alice enviar seu número de cartão de crédito para 'Bob', a fim de adquirir algumas mercadorias.

Uma desvantagem desse clássico ataque do homem no meio é que Trudy tem de estar em uma posição conveniente para interceptar o tráfego enviado por Alice e forjar seu tráfego de entrada. Na prática, ela tem de grampear a linha telefônica de Alice ou de Bob, pois é muito difícil grampear o backbone de fibra óptica. Embora a espionagem ativa certamente seja possível, ela exige determinado volume de trabalho e, embora seja inteligente, Trudy também é preguiçosa. Além disso, existem maneiras mais fáceis de enganar Alice.

DNS spoofing

Suponha que Trudy seja capaz de invadir o sistema DNS, ou talvez apenas o cache DNS no ISP de Alice, e substitua o endereço IP de Bob (digamos, 36.1.2.3) por seu próprio endereço IP (digamos, 42.9.9.9). Isso leva ao ataque explicado a seguir. O modo como ele deve funcionar é ilustrado na Figura 8.42(a). Aqui (1) Alice solicita ao DNS o endereço IP de Bob, (2) recebe esse endereço, (3) pergunta a Bob por sua home page e (4) também recebe a home page. Depois de Trudy ter modificado o registro de DNS de Bob para conter seu próprio endereço IP em lugar do endereço de Bob, temos a situação da Figura 8.42(b). Nesse caso, quando Alice procurar o endereço IP de Bob, receberá o de Trudy, e então todo o tráfego destinado a Bob irá para Trudy. Agora, Trudy pode montar um ataque do homem no meio sem ter o trabalho de grampear linhas telefônicas. Em vez disso, ela terá de invadir um servidor DNS e alterar um único registro, uma proposta muito mais fácil.

Como Trudy poderia enganar o DNS? Na verdade, isso é relativamente fácil. Em resumo, Trudy pode enganar o servidor DNS no ISP de Alice, enviando-lhe uma consulta do endereço IP de Bob. Infelizmente, como o DNS utiliza o UDP, o servidor DNS não tem nenhum meio real de verificar quem forneceu a resposta. Trudy pode explorar essa propriedade forjando a resposta esperada e, desse modo, injetar um falso endereço IP no cache do servidor DNS. Por simplicidade, vamos supor que o ISP de Alice não tenha uma entrada para o site de Bob, bob.com. Se tiver, Trudy poderá esperar até ele entrar em timeout e tentar mais tarde (ou usar outros truques).

Trudy inicia o ataque enviando uma solicitação de pesquisa ao ISP de Alice, pedindo o endereço IP de bob.com. Tendo em vista que não existe nenhuma entrada correspondente a esse nome no DNS, o servidor cache consulta o servidor de nível superior em busca do domínio *com*, a fim de obter uma entrada. No entanto, Trudy invade o servidor *com* e envia de volta uma resposta falsa que informa: "*bob.com* é 42.9.9.9", mas, na realidade, o endereço IP é o dela. Se sua falsa resposta voltar primeiro ao ISP de Alice, ela será guardada no cache e a resposta real será rejeitada como uma resposta não solicitada a uma consulta que não está mais pendente. Enganar um servidor DNS fazendo-o instalar um falso endereço IP é uma ação chamada **DNS spoofing**. Um cache que contém um endereço IP intencionalmente falso como esse é chamado **cache envenenado**.

Na realidade, nem tudo é assim tão simples. Primeiro, o ISP de Alice verifica se a resposta contém o endereço IP de origem correto do servidor de nível superior. Porém, como Trudy pode colocar o que quiser nesse campo de endereço IP, ela pode anular com facilidade esse teste, pois os endereços IP dos servidores de nível superior têm de ser públicos.

Em segundo lugar, para permitir que os servidores DNS saibam a resposta para cada solicitação, todas as solicitações têm um número de sequência. Para enganar o ISP de Alice, Trudy tem de conhecer seu número de sequência atual. O modo mais fácil de descobrir o número de sequência atual é Trudy registrar um domínio ela mesma, digamos, trudy-a-intrusa.com. Vamos supor que seu endereço IP também

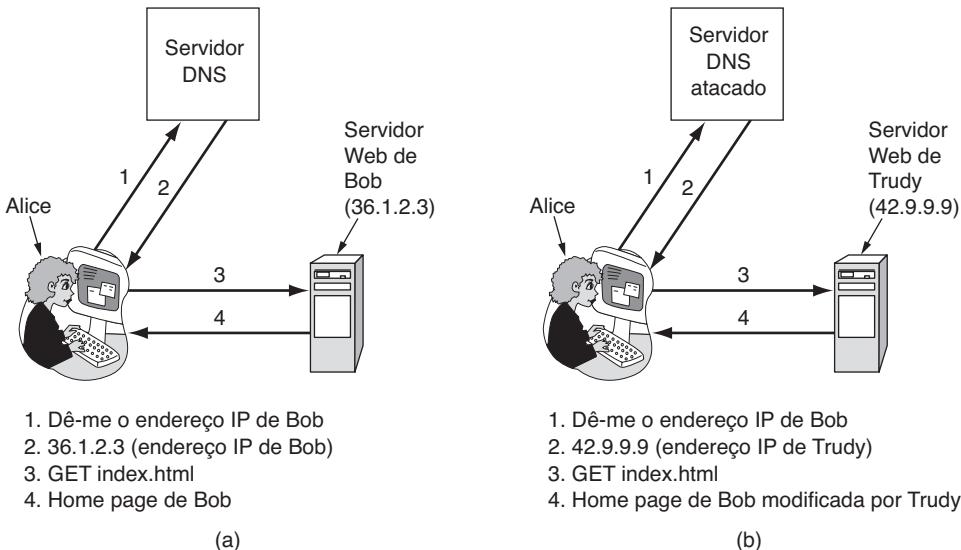
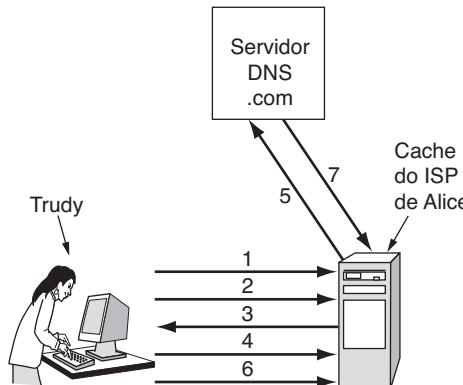



Figura 8.42 | (a) Situação normal. (b) Um ataque baseado na invasão do DNS e na modificação do registro de Bob.

seja 42.9.9.9. Ela também cria um servidor DNS para seu novo domínio, dns.trudy-a-intrusa.com. Esse servidor utiliza ainda o endereço IP de Trudy, 42.9.9.9, pois Trudy só tem um computador. Agora, ela tem de dar ciência ao ISP de Alice de seu servidor DNS. Isso é fácil. Tudo o que tem a fazer é pedir ao ISP de Alice que lhe forneça foobar.trudy-a-intrusa.com. Isso fará com que o ISP de Alice descubra quem serve ao novo domínio de Trudy, solicitando o endereço ao servidor de endereços *com* de nível superior.

Com dns.trudy-a-intrusa.com em segurança no cache do ISP de Alice, o ataque real pode começar. Agora, Trudy consulta o ISP de Alice em busca de www.trudy-a-intrusa.com. Naturalmente, o ISP envia ao servidor DNS de Trudy uma consulta solicitando essa página. Essa consulta contém o número de sequência que Trudy está procurando. Mais que depressa, Trudy pede ao ISP de Alice que procure Bob. Ela responde de imediato sua própria pergunta, enviando ao ISP uma resposta forjada, supostamente do servidor *com* de nível superior, afirmando: 'bob.com é 42.9.9.9'. Essa resposta contém um número de sequência uma unidade maior que o endereço que ela acabou de receber. Enquanto isso, ela também pode enviar uma segunda falsificação com um número de sequência duas unidades mais alto, e talvez mais uma dezena de números de sequência crescentes. Um deles deverá corresponder. Os restantes serão simplesmente descartados. Quando chegar a resposta forjada de Alice, ela ficará em cache; mais tarde, quando chegar a resposta real, ela será rejeitada, pois não haverá nenhuma consulta pendente nesse momento.

Agora, quando Alice procurar bob.com, será informada de que deve usar 42.9.9.9, o endereço de Trudy. No conforto de sua própria sala de estar, Trudy montou um ataque do homem no meio bem-sucedido, cujas etapas estão ilustradas na Figura 8.43. Esse ataque específico pode ser anulado se os servidores DNS usarem IDs aleatórias em suas consultas, em vez de simplesmente contarem; porém, parece que toda vez que um furo é vedado, surge outro. Em particular, as IDs têm apenas 16 bits, de modo que é fácil trabalhar com todas elas quando é um computador que está fazendo as tentativas.

DNS Seguro

O problema real é que o DNS foi projetado em uma época na qual a Internet era um recurso de pesquisa para algumas centenas de universidades e nem Alice, nem Bob, nem Trudy faziam parte da turma. Naquele tempo, o importante não era a segurança, mas sim fazer a Internet funcionar. O ambiente mudou de forma radical ao longo dos anos; assim, em 1994, a IETF instalou um grupo de trabalho para tornar o DNS fundamentalmente seguro. Esse projeto (contínuo) é conhecido como **DNSsec (DNS security)**, e seu resultado é apresentado na RFC 2535. Infelizmente, o DNSsec ainda não foi totalmente implementado, e portanto diversos servidores DNS ainda estão vulneráveis a ataques de spoofing.

Em termos conceituais, o DNSsec é extremamente simples. Ele se baseia na criptografia de chave pública. Cada zona DNS (no sentido da Figura 7.2) tem um par de chave pública/chave privada. Todas as informações enviadas por um servidor DNS são assinadas com a chave privada da zona de origem, de forma que o receptor possa verificar sua autenticidade.

O DNSsec oferece três serviços fundamentais:

1. prova de onde os dados se originaram;
2. distribuição de chave pública;
3. autenticação de transação e solicitação.

1. Procura foobar.trudy-the-intruder.com (para forçá-lo para o cache do ISP)
2. Procura www.trudy-the-intruder.com (para obter o próximo número de sequência do ISP)
3. Pedido para www.trudy-the-intruder.com (para forçar o ISP a consultar o servidor .com na etapa 5)
4. Rapidamente, procura bob.com (para forçar o ISP a consultar o servidos no passo 5)
5. Consulta legítima para bob.com com seq = n+1
6. Resposta forjada de Trudy: Bob é 42.9.9.9, seq = n+1
7. Resposta real (rejeitada; tarde demais)

Figura 8.43 | Como Trudy engana o ISP de Alice.

O principal serviço é o primeiro, que verifica se os dados que estão sendo retornados foram aprovados pelo proprietário da zona. O segundo é útil para armazenar e recuperar chaves públicas com segurança. O terceiro é necessário como proteção contra ataques por reprodução e spoofing. Observe que o sigilo não é um serviço oferecido, pois todas as informações no DNS são consideradas públicas. Tendo em vista que a implantação do DNSsec deverá demorar vários anos, a habilidade de servidores conscientes da segurança para interoperar com servidores que ignoram os aspectos da segurança é algo essencial; isso implica que o protocolo não pode ser alterado. Agora, vamos observar alguns detalhes.

Os registros DNS são agrupados em conjuntos chamados **RRSets (Resource Record Sets)**, com todos aqueles que têm o mesmo nome, a mesma classe e o mesmo tipo reunidos em um único conjunto. Por exemplo, um RRSet pode conter vários registros *A*, se um nome DNS for resolvido em um endereço IP primário e um endereço IP secundário. Os RRSets são estendidos com vários tipos novos de registros (descritos a seguir). Cada RRSet passa por um hash criptográfico (por exemplo, usando SHA-1). O hash é assinado pela chave privada da zona (por exemplo, usando-se o RSA). A unidade de transmissão para clientes é o RRSet assinado. Ao receber um RRSet assinado, o cliente pode verificar se ele foi assinado pela chave privada da zona de origem. Se a assinatura coincidir, os dados serão aceitos. Tendo em vista que cada RRSet contém sua própria assinatura, os RRSets podem ser armazenados em cache em qualquer lugar, até mesmo em servidores não confiáveis, sem trazer perigo à segurança.

O DNSsec introduz vários tipos novos de registros. O primeiro deles é o registro *KEY*. Esse registro contém a chave pública de uma zona, um usuário, um host ou outro protagonista, o algoritmo de criptografia usado na assinatura, o protocolo empregado na transmissão e alguns outros bits. A chave pública é armazenada em estado bruto. Os certificados X.509 não são usados devido a seu tamanho. O campo do algoritmo contém um valor 1 para assinaturas MD5/RSA (a opção preferida) e outros valores para outras combinações. O campo de protocolo pode indicar o uso do IPsec ou de outros protocolos de segurança, se houver.

O segundo entre os novos tipos de registros é o registro *SIG*. Ele contém o hash assinado de acordo com o algoritmo especificado no registro *KEY*. A assinatura se aplica a todos os registros no RRSet, incluindo quaisquer registros *KEY* presentes, mas excluindo ela própria. Ele também contém os horários em que a assinatura inicia seu período de validade e de vencimento, bem como o nome do signatário e de alguns outros itens.

O projeto do DNSsec é tal que a chave privada de uma zona pode ser mantida off-line. Uma ou duas vezes por dia, o conteúdo do banco de dados de uma zona pode ser transportado manualmente (por exemplo, em CD-ROM) para uma máquina desconectada na qual a chave privada está localizada. Todos os RRSets podem ser assinados nessa máquina e os registros SIG assim produzidos podem ser transportados de volta ao servidor primário da zona em CD-ROM. Desse modo, a chave privada pode ser armazenada em um CD-ROM bloqueado de forma segura, exceto quando é inserido na máquina desconectada para assinar os novos RRSets do dia. Depois que o processo de assinatura é concluído, todas as cópias da chave são apagadas da memória, sendo o disco e o CD-ROM devolvidos a um local seguro. Esse procedimento reduz a segurança eletrônica à segurança física, algo que as pessoas entendem como tratar.

Esse método de assinatura prévia de RRSets aumenta bastante a velocidade do processo de responder a consultas, pois nenhuma criptografia tem de ser feita durante a execução. Em compensação, é necessário um grande volume de espaço de disco para armazenar todas as chaves e assinaturas nos bancos de dados DNS. Alguns registros aumentarão dez vezes em tamanho devido à assinatura.

Quando um processo cliente obtém um RRSet assinado, ele tem de aplicar a chave pública da zona de origem

para decifrar o hash, calcular o próprio hash e comparar os dois valores. Se eles concordarem, os dados serão considerados válidos. Porém, esse procedimento faz surgir a seguinte questão: como o cliente obtém a chave pública da zona? Uma alternativa é adquiri-la de um servidor confiável, utilizando uma conexão segura (por exemplo, usando o IPsec).

Porém, na prática, espera-se que os clientes sejam pré-configurados com as chaves públicas de todos os domínios de nível superior. Se agora Alice quiser visitar o site de Bob, ela poderá solicitar ao DNS o RRSet de bob.com, que conterá seu endereço IP e um registro *KEY* contendo a chave pública de Bob. Esse RRSet será assinado pelo domínio *com* de nível superior, e assim Alice poderá verificar facilmente sua validade. Um exemplo do que esse RRSet pode conter é mostrado na Tabela 8.4.

Agora, munida de uma cópia verificada da chave pública de Bob, Alice pode pedir ao servidor DNS de Bob (controlado por Bob) o endereço IP de www.bob.com. Esse RRSet será assinado pela chave privada de Bob, e assim Alice pode verificar a assinatura de Bob no RRSet que ele retorna. Se Trudy, de alguma maneira, conseguir injetar um falso RRSet em qualquer dos caches, Alice poderá detectar essa falta de autenticidade facilmente, porque o registro *SIG* contido nele será incorreto.

Porém, o DNSsec também fornece um mecanismo criptográfico para vincular uma resposta a uma consulta específica, a fim de impedir o tipo de ataque que Trudy tentou realizar na Figura 8.43. Essa medida (opcional) contra o spoofing adiciona à resposta um hash da mensagem de consulta assinado com a chave privada do autor da resposta. Como Trudy não conhece a chave privada do servidor *com* de nível superior, ela não pode forjar uma resposta a uma consulta ao ISP de Alice enviada pelo ISP. Sem dúvida, ela pode receber sua resposta de volta primeiro, mas essa resposta será rejeitada devido à assinatura inválida do hash.

O DNSsec também admite alguns outros tipos de registros. Por exemplo, o registro *CERT* pode ser usado para armazenar certificados (por exemplo, X.509). Esse registro é fornecido porque algumas pessoas querem transformar o DNS em uma PKI. Resta saber se de fato isso é possível. Interromperemos nossa discussão sobre o DNSsec aqui. Para obter mais detalhes, consulte a RFC 2535.

8.9.3 SSL — Secure Sockets Layer

A nomenclatura segura é um bom começo, mas existem vários outros detalhes sobre segurança da Web. A próxima etapa é gerar conexões seguras. Agora, vamos examinar como podem ser alcançadas. Nada que envolva segurança é simples, e isto não é uma exceção.

Quando estourou para o público, a Web foi usada no início apenas para distribuir páginas estáticas. Porém, em pouco tempo, algumas empresas tiveram a ideia de usá-la para transações financeiras, como a compra de mercadorias por cartões de crédito, transações bancárias on-line e mercado de capitais eletrônico. Essas aplicações criaram uma demanda por conexões seguras. Em 1995, a Netscape Communications Corp., que então dominava o mercado de fabricantes de navegadores, respondeu introduzindo um pacote de segurança chamado **SSL (Secure Sockets Layer)** para atender a essa demanda. Esse software e seu protocolo agora também são amplamente utilizados, por exemplo, por Firefox, Safari e Internet Explorer, e portanto vale a pena examiná-los com mais detalhes.

O SSL constrói uma conexão segura entre dois soquetes, incluindo:

1. negociação de parâmetros entre cliente e servidor.
2. autenticação mútua de cliente e servidor.
3. comunicação secreta.
4. proteção da integridade dos dados.

Já vimos esses itens antes e, portanto, não há necessidade de desenvolvê-los.

O posicionamento do SSL na pilha de protocolos habitual é ilustrado na Figura 8.44. Efetivamente, trata-se de uma nova camada colocada entre a de aplicação e a de transporte, aceitando solicitações do navegador e enviando-as ao TCP para transmissão ao servidor. Depois que a conexão segura é estabelecida, a principal tarefa do SSL é manipular a compactação e a criptografia. Quando o HTTP é usado sobre SSL, ele se denomina **HTTPS (Secure HTTP)**, embora seja o protocolo HTTP padrão. Às vezes, ele está disponível em uma nova porta (443), em lugar da porta-padrão (80). A propósito, SSL não se limita ao uso apenas com navegadores da Web, mas essa é sua aplicação mais comum. Ele também pode oferecer autenticação mútua.

Nome de domínio	Tempo de vida	Classe	Tipo	Valor
bob.com.	86400	IN	A	36.1.2.3
bob.com.	86400	IN	KEY	3682793A7B73F731029CE2737D...
bob.com.	86400	IN	SIG	86947503A8B848F5272E53930C...

Tabela 8.4 | Um exemplo de RRSet para bob.com. O registro KEY é a chave pública de Bob. O registro *SIG* é o hash assinado do servidor *com* de nível superior dos registros *A* e *KEY*, a fim de verificar sua autenticidade.

Aplicação (HTTP)
Segurança (SSL)
Transporte (TCP)
Rede (IP)
Enlace de dados (PPP)
Física (modem, ADSL, TV a cabo)

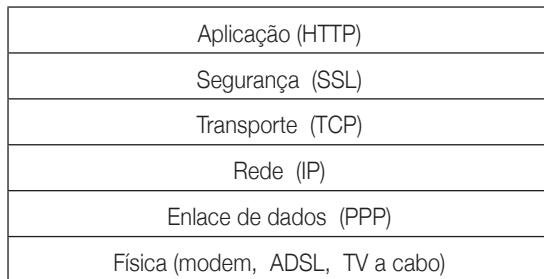



Figura 8.44 | Camadas (e protocolos) para um usuário doméstico navegando com SSL.

O protocolo SSL passou por várias versões. Descreveremos apenas a versão 3, que é a mais utilizada. O SSL admite uma variedade de opções, que incluem a presença ou a ausência de compactação, os algoritmos criptográficos a ser usados e algumas questões relativas a restrições de exportação impostas à criptografia. A última se destina principalmente a assegurar que a criptografia é utilizada apenas quando ambas as extremidades da conexão estão nos Estados Unidos. Em outros casos, as chaves serão limitadas a 40 bits, que os criptógrafos consideram uma piada. A Netscape foi forçada a colocar essa restrição para obter uma licença de exportação do governo dos Estados Unidos.

O SSL consiste em dois subprotocolos, um para estabelecer uma conexão segura e outro para usá-la. Vamos começar examinando como as conexões seguras são estabelecidas. O subprotocolo de estabelecimento de conexões é mostrado na Figura 8.45. Ele começa com a mensagem 1, quando Alice envia uma solicitação a Bob para estabelecer uma conexão. A solicitação especifica a versão do SSL que Alice tem e suas preferências com relação aos algoritmos de compactação e de criptografia. Ela também contém um nonce R_A, a ser usado mais tarde.

Agora é a vez de Bob. Na mensagem 2, Bob escolhe entre os diversos algoritmos que Alice pode admitir e envia seu próprio nonce, R_B. Em seguida, na mensagem 3, ele envia um certificado contendo sua chave pública. Se esse certificado não for assinado por alguma autoridade conhecida, ele também envia uma cadeia de certificados que pode ser seguida de volta até chegar a uma autoridade original. Todos os navegadores, inclusive o de Alice, são pré-carregados com cerca de 100 chaves públicas; assim, se Bob puder estabelecer uma cadeia ancorada em uma dessas chaves, Alice será capaz de verificar a chave pública de Bob. Nesse momento, Bob pode enviar algumas outras mensagens (como uma solicitação do certificado de chave pública de Alice). Ao terminar, Bob envia a mensagem 4 para dizer a Alice que agora é a vez dela.

Alice responde escolhendo ao acaso uma chave **pré-mestre** aleatória de 384 bits e a envia para Bob, codificada com a chave pública de Bob (mensagem 5). A chave de sessão real usada para codificar os dados é derivada da chave pré-mestre combinada com ambos os nonces de modo complexo. Depois que a mensagem 5 é recebida, Alice e Bob são capazes de calcular a chave de sessão. Por essa razão, Alice informa a Bob que ele deve passar para a nova cifra (mensagem 6) e também que ela concluiu o subprotocolo de estabelecimento (mensagem 7). Bob então confirma as mensagens de Alice (mensagens 8 e 9).

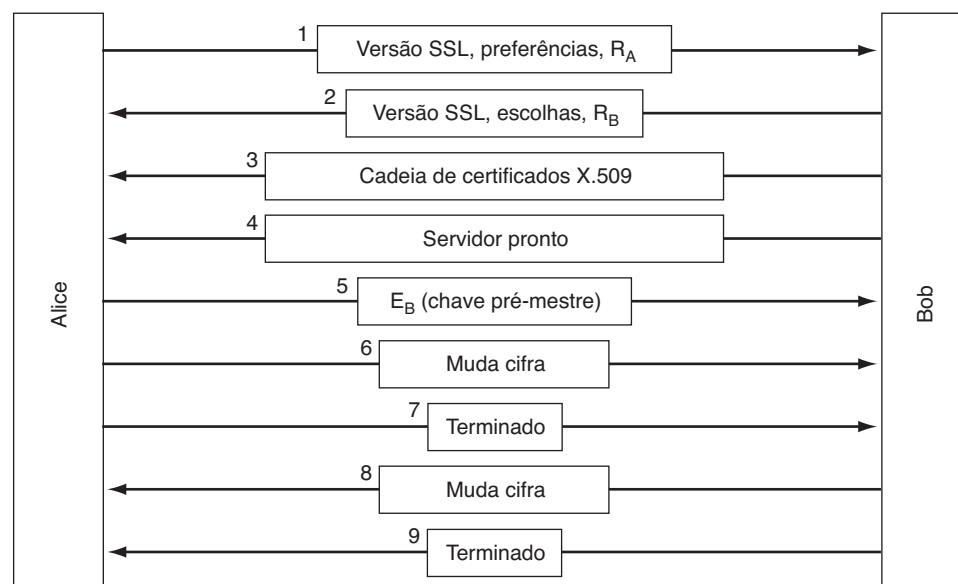



Figura 8.45 | Uma versão simplificada do subprotocolo de estabelecimento de conexões SSL.

Porém, embora Alice saiba quem é Bob, ele não sabe quem é Alice (a menos que ela tenha uma chave pública e um certificado correspondente, uma situação improvável para um indivíduo). Portanto, a primeira mensagem de Bob pode ser uma solicitação para Alice se conectar usando um nome de login e uma senha estabelecidos anteriormente. No entanto, o protocolo de login está fora do escopo do SSL. Depois que ele é realizado por quaisquer meios, o transporte de dados pode se iniciar.

Como mencionamos antes, o SSL admite vários algoritmos de criptografia. O mais forte usa o DES triplo com três chaves separadas para criptografia e com o SHA-1, a fim de manter a integridade das mensagens. Essa combinação é relativamente lenta, e assim é usada principalmente em aplicações bancárias e outras aplicações nas quais é exigida a mais alta segurança. Para aplicações comuns de comércio eletrônico, é usado o RC4 com uma chave de 128 bits para criptografia, e o MD5 é empregado para autenticação de mensagens. O RC4 utiliza a chave de 128 bits como semente e a expande até um número muito maior para uso interno. Em seguida, ele usa esse número interno para gerar um fluxo de chaves. O fluxo de chaves é submetido a uma operação XOR com texto simples para fornecer um fluxo de cifras clássico, como vimos na Figura 8.12. As versões de exportação também utilizam o RC4 com chaves de 128 bits, mas 88 dos bits são divulgados ao público para facilitar a violação da cifra.

Para o transporte real, é usado um segundo subprotocolo, como mostra a Figura 8.46. Primeiro, as mensagens do navegador são divididas em unidades de até 16 KB. Se a compactação estiver ativa, cada unidade será então compactada em separado. Depois disso, uma chave secreta derivada dos dois nonces e da chave pré-mestre é concatenada com o texto compactado, e o resultado passa por um hash com o algoritmo de hash combinado (normalmente, o MD5). Esse hash é anexado a cada fragmento como o MAC. O fragmento compactado somado ao MAC é então codificado com o algoritmo de criptografia simétrica estabelecido de comum acordo (em geral, por uma operação XOR entre ele e o fluxo de chaves do RC4). Por fim, é anexado o cabeçalho do fragmento que é transmitido pela conexão TCP.

No entanto, é importante um alerta. Por ter sido mostrado que o RC4 tem algumas chaves fracas, que podem ser facilmente criptoanalisadas, a segurança do SSL usando o RC4 é precária. (Fluhrer et al., 2001). Os navegadores que permitem ao usuário escolher o conjunto de cifras devem ser configurados para usar o DES triplo com chaves de 168 bits e o SHA-1 o tempo todo, embora essa combinação seja mais lenta que o RC4 e o MD5. Ou, melhor ainda, os usuários devem fazer o upgrade para navegadores que ofereçam suporte ao sucessor do SSL, que descreveremos em breve.

Um problema com o SSL é que os protagonistas não podem ter certificados e, mesmo que tenham, nem sempre verificam se as chaves que estão sendo usadas correspondem aos certificados. Em 1996, a Netscape Communications Corp. submeteu o SSL à IETF para padronização. O resultado foi o **TLS** (**Transport Layer Security**), descrito na RFC 5246.

O TLS foi embutido no SSL versão 3. As mudanças feitas no SSL foram relativamente pequenas, mas suficientes para o SSL versão 3 e o TLS não conseguirem interoperar. Por exemplo, o modo como a chave de sessão é derivada da chave pré-mestre e dos nonces mudou para tornar a chave mais forte (isto é, mais difícil de ser violada por criptoanálise). Devido à incompatibilidade, a maioria dos navegadores implementa os dois protocolos, com o TLS passando para SSL durante a negociação, se for necessário. Isso é conhecido

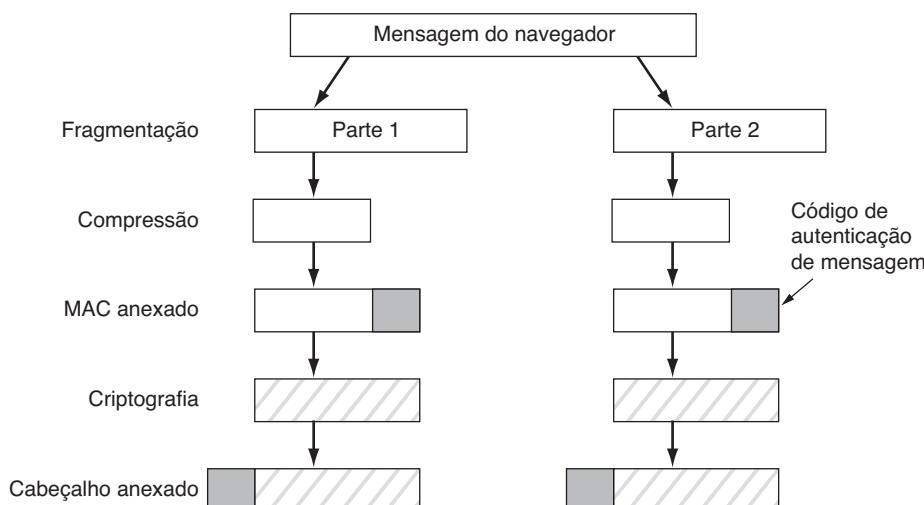



Figura 8.46 | Transmissão de dados com SSL.

como SSL/TLS. A primeira implementação do TLS apareceu em 1999 com a versão 1.2 definida em agosto de 2008. Ela inclui suporte para conjuntos de cifras mais fortes (principalmente AES). O SSL continuou sendo forte no mercado, embora o TLS provavelmente o substituísse aos poucos.

8.9.4 Segurança em código móvel

A nomenclatura e as conexões são duas áreas de preocupação relacionadas à segurança da Web, mas existem outras. No início, quando as páginas Web eram apenas arquivos HTML estáticos, elas não continham código executável. Agora, as páginas frequentemente contêm pequenos programas, inclusive miniaplicativos Java, controles ActiveX e JavaScripts. Baixar e executar esse **código móvel** é sem dúvida um grande risco de segurança; por essa razão, foram criados vários métodos para minimizá-lo. Agora, veremos rapidamente algumas questões geradas pelo código móvel e algumas abordagens para lidar com ele.

Segurança em miniaplicativos Java

Os miniaplicativos (applets) Java são pequenos programas em Java, compilados para uma linguagem de máquina orientada a pilhas de instruções (stacks), chamada **JVM (Java Virtual Machine)**. Eles podem ser colocados em uma página Web para ser transferidos por download juntamente com a página. Depois que a página é carregada, os miniaplicativos são inseridos em um interpretador JVM no navegador, como ilustra a Figura 8.47.

A vantagem de executar código interpretado sobre código compilado é que cada instrução é examinada pelo interpretador antes de ser executada. Isso dá ao interpretador a oportunidade de verificar se o endereço da instrução é válido. Além disso, as chamadas do sistema também são capturadas e interpretadas. A forma como essas chamadas são manipuladas é um assunto relacionado à política de segurança. Por exemplo, se um miniaplicativo for confiável

Figura 8.47 | Os miniaplicativos podem ser interpretados por um navegador Web.

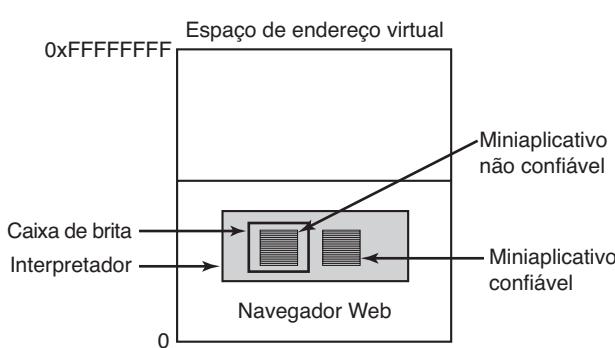
(como no caso de ele vir do disco local), suas chamadas do sistema poderão ser executadas sem questionamentos. Porém, se um miniaplicativo não for confiável (por exemplo, se veio da Internet), ele poderá ser encapsulado em uma **caixa de brita** (sandbox) para limitar seu comportamento e bloquear suas tentativas de usar recursos do sistema.

Quando um miniaplicativo tenta usar um recurso do sistema, sua chamada é repassada a um monitor de segurança para aprovação. O monitor examina a chamada levando em conta a política de segurança local e depois toma a decisão de permiti-la ou rejeitá-la. Desse modo, é possível dar aos miniaplicativos acesso a alguns recursos, mas não a todos. Infelizmente, a realidade é que o modelo de segurança funciona mal e que os bugs que ele contém surgem o tempo todo.

ActiveX

Os controles ActiveX são programas binários do x86 que podem ser incorporados às páginas Web. Quando um deles é encontrado, é realizada uma verificação para saber se deve ser executado e, se passar no teste, assim é feito. Um controle ActiveX não é interpretado ou colocado em uma caixa de brita de forma alguma; portanto, tem tanto poder como qualquer outro programa do usuário e tem potencial para causar grandes danos. Desse modo, toda a segurança se resume à decisão de executar ou não o controle ActiveX. Concluindo, a ideia geral é um furo de segurança gigantesco.

O método que a Microsoft escolheu para tomar essa decisão se baseia na ideia de **assinatura de código**. Cada controle ActiveX é acompanhado por uma assinatura digital — um hash do código assinado por seu criador com o uso de criptografia de chave pública. Quando um controle ActiveX aparece, primeiro o navegador verifica a assinatura para ter certeza de que ele não foi adulterado em trânsito. Se a assinatura estiver correta, o navegador consulta suas tabelas internas para ver se o criador do programa é confiável, ou se existe uma cadeia de confiança que leve de volta até um criador confiável. Se o criador for confiável, o programa será executado; caso contrário, não. O sistema da Microsoft para verificar controles ActiveX é chamado **Authenticode**.

É útil comparar as abordagens Java e ActiveX. Com a abordagem Java, não é feita nenhuma tentativa para determinar quem escreveu o miniaplicativo. Em vez disso, um interpretador runtime certifica-se de que ele não executa ações que o proprietário da máquina afirmou que os miniaplicativos não poderiam executar. Em contraste, no caso da assinatura de código, não é feita nenhuma tentativa de monitorar o comportamento de runtime do código móvel. Se veio de uma origem confiável e não foi modificado em trânsito, ele simplesmente é executado. Não há nenhuma tentativa de verificar se o código é malicioso ou não. Se era intenção do programador original criar um código que formatasse o disco rígido e depois apagasse a ROM flash

para que o computador nunca mais pudesse ser iniciado, isso não importa: se o programador foi certificado como confiável, o código será executado e destruirá o computador (a menos que os controles ActiveX estejam desativados no navegador).

Muitas pessoas têm receio de confiar em uma empresa de software desconhecida. Para demonstrar o problema, um programador em Seattle formou uma empresa de software e conseguiu certificação como origem fidedigna, o que é fácil. Em seguida, desenvolveu um controle ActiveX que provocava um desligamento limpo da máquina e distribuiu amplamente seu controle ActiveX. Ele desativou muitas máquinas, mas elas podiam simplesmente ser reiniciadas, e, portanto, não havia nenhum dano. O programador estava apenas tentando expor o problema ao mundo. A resposta oficial foi revogar o certificado para esse controle ActiveX, o que encerrou um breve episódio de forte embaraço, mas o problema subjacente ainda persiste, à espera de que um programador terrível o explore (Garfinkel com Spafford, 2002). Tendo em vista que não há como policiar milhares de empresas de software que poderiam escrever código móvel, a técnica de assinatura de código é um desastre que pode ocorrer a qualquer momento.

JavaScript

O JavaScript não tem nenhum modelo de segurança formal, mas um longo histórico de implementações inseguras. Cada fornecedor cuida da segurança de um modo diferente. Por exemplo, a versão 2 do Netscape Navigator usava algo similar ao modelo do Java mas, na versão 4, essa estratégia foi abandonada em favor de um modelo de assinatura de código.

O problema fundamental é que permitir a execução de código estranho em sua máquina significa procurar problemas. Do ponto de vista da segurança, seria como convidar um assaltante a entrar em sua casa, e depois tentar observá-lo com cuidado para que ele não possa escapar da cozinha para a sala de estar. Se acontecer algo inesperado e você estiver distraído por um momento, poderão surgir consequências desastrosas. A tensão aqui é o fato de que o código móvel permite imagens gráficas atraentes e interação rápida, e muitos projetistas de sites acham que isso é muito mais importante que a segurança, especialmente quando é a máquina de outra pessoa que está correndo riscos.

Extensões do navegador

Além de estender as páginas Web com código, existe um mercado em explosão nas **extensões do navegador**, **suplementos** e **plug-ins**. Eles são programas de computador que estendem a funcionalidade dos navegadores Web. Os plug-ins normalmente oferecem a capacidade de interpretar ou exibir um certo tipo de conteúdo, como PDFs ou animações Flash. Extensões e suplementos oferecem novos recursos ao navegador, como melhor gerenciamento de senhas ou formas de interagir com páginas, por exemplo, marcando-as ou facilitando a compra para itens relacionados.

Instalar uma extensão, suplemento ou plug-in é tão simples quanto encontrar algo que você deseja ao navegar e selecionar o link para instalar o programa. Essa ação fará o código ser baixado pela Internet e instalado no navegador. Todos esses programas são escritos para frameworks que diferem dependendo do navegador que está sendo melhorado. Porém, de certa forma, eles se tornam parte da base de computação confiável do navegador. Ou seja, se o código que for instalado tiver bugs, o navegador inteiro pode ser comprometido.

Também existem dois outros modos de falha óbvios. O primeiro é que o programa pode se comportar de forma maliciosa, por exemplo, reunindo informações pessoais e enviando-as para um servidor remoto. Por tudo o que o navegador sabe, o usuário instalou a extensão exatamente para essa finalidade. O segundo problema é que os plug-ins dão ao navegador a capacidade de interpretar novos tipos de conteúdo. Frequentemente, esse conteúdo é uma linguagem de programação completa por si só. PDF e Flash são bons exemplos. Quando os usuários exibem páginas com conteúdo PDF ou Flash, os plug-ins em seu navegador estão executando o código PDF e Flash. É melhor que esse código seja seguro; costuma haver vulnerabilidades que se pode explorar. Por todas essas razões, os suplementos e plug-ins só devem ser instalados se realmente forem necessários e apenas de fornecedores confiáveis.

Vírus

Os vírus são outra forma de código móvel. Porém, diferentemente dos exemplos anteriores, os vírus sempre chegam sem ser convidados. A diferença entre um vírus e o código móvel comum é que o vírus é desenvolvido para se reproduzir. Quando um vírus chega, através de uma página, de um anexo de correio eletrônico ou de algum outro modo, em geral ele começa infectando programas executáveis no disco. Quando um desses programas é executado, o controle é transferido para o vírus, que, em geral, tenta se difundir para outras máquinas, por exemplo, enviando cópias de si mesmo por correio eletrônico para todas as pessoas que têm seu nome no catálogo de endereços da vítima. Alguns vírus infectam o setor de inicialização do disco rígido; assim, quando a máquina é inicializada, o vírus é executado. Os vírus se tornaram um problema enorme na Internet e causam prejuízos de bilhões de dólares. Não existe nenhuma solução óbvia. Talvez uma nova geração de sistemas operacionais, inteiramente baseada em microkernels seguros e rígida divisão dos usuários, processos e recursos em compartimentos estanques possa ajudar a resolver o problema.

8.10 Questões sociais

A Internet e sua tecnologia de segurança compoõem uma área para a qual convergem as questões sociais, a política pública e a tecnologia, não raro com enormes consequências. Apresentaremos a seguir um breve exame de três áreas: privacidade, liberdade de expressão e direito autoral. É desnecessário dizer que trataremos o assunto de maneira superficial. Para leitura adicional, consulte Anderson (2008a), Garfinkel com Spafford (2002) e Schneier (2004). A Internet também está repleta de material sobre o tema. Basta digitar palavras como 'privacidade', 'censura' e 'direito autoral' em qualquer mecanismo de pesquisa.

8.10.1 Privacidade

As pessoas têm direito à privacidade? Boa pergunta. A Quarta Emenda da Constituição dos Estados Unidos proíbe o governo de realizar buscas na casa das pessoas, vasculhar seus documentos e seus bens sem uma boa razão, e continua a restringir as circunstâncias sob as quais devem ser emitidos os mandados de busca. Desse modo, a privacidade é um direito público há mais de 200 anos, pelo menos nos Estados Unidos.

O que mudou na última década foi a facilidade com que os governos podem espionar seus cidadãos e a facilidade com que os cidadãos podem impedir tais atos de espionagem. No século XVIII, para o governo realizar buscas nos documentos de um cidadão, ele tinha de enviar um policial a cavalo até a sua fazenda, exigindo a apresentação de certos documentos. Era um procedimento incômodo. Hoje em dia, as companhias telefônicas e os provedores da Internet fornecem prontamente grampos ao receber mandados de busca. Isso facilita a vida do policial e ele não corre o risco de cair do cavalo.

A criptografia muda tudo isso. Qualquer pessoa que se dê ao trabalho de baixar e instalar o PGP e que utilize uma chave com força de alienígena bem protegida pode ter certeza de que ninguém no universo conhecido poderá ler sua correspondência de correio eletrônico, com ou sem mandado de busca. Os governos entendem bem esse problema e não o apreciam. A privacidade real significa que é muito mais difícil seus agentes espionarem criminosos de todos os tipos, mas também é muito mais difícil espionar jornalistas e adversários políticos. Como resultado, alguns governos restringem ou proíbem o uso ou a exportação de criptografia. Por exemplo, na França, antes de 1999, toda criptografia era proibida, a menos que o governo recebesse as chaves.

A França não estava só. Em abril de 1993, o governo dos Estados Unidos anunciou sua intenção de criar um criptoprocessador em hardware, o **clipper chip**, o padrão para todas as comunicações em rede. Desse modo, diziam, a privacidade dos cidadãos estaria garantida. Ele também mencionava que o chip fornecia ao governo a possibilidade de decodificar todo tráfego por meio de um esquema chamado **custódia de chaves**, que permitia o acesso do governo a todas as chaves. No entanto, ele prometia só espiar quando tivesse um mandado de busca válido. Não é preciso dizer que o resultado foi uma enorme agitação, com os defensores da privacidade denunciando todo o plano e os promotores de justiça elogiando o esquema. Por fim, o governo voltou atrás e descartou a ideia.

Há um grande volume de informações sobre privacidade eletrônica disponível no site da Electronic Frontier Foundation, em www.eff.org.

Repostadores anônimos

PGP, SSL e outras tecnologias tornam possível duas partes estabelecerem comunicação segura e autenticada, livre de vigilância e interferência de terceiros. Porém, às vezes a privacidade é mais bem servida quando não há autenticação, tornando a comunicação anônima. O anonimato pode ser interessante em mensagens ponto a ponto, newsgroups ou ambos.

Vamos considerar alguns exemplos. Primeiro, dissidentes políticos que vivem em regimes autoritários muitas vezes desejam se comunicar de forma anônima para escapar da prisão ou de ser assassinados. Segundo, delitos em muitas organizações corporativas, educacionais, governamentais e outras frequentemente são expostos por delatores, que muitas vezes preferem permanecer anônimos para evitar represálias. Terceiro, pessoas com visões sociais, políticas ou religiosas impopulares podem desejar se comunicar umas com as outras por correio eletrônico ou newsgroups, sem se expor. Em quarto lugar, as pessoas podem desejar discutir alcoolismo, doenças mentais, abusos sexuais, pedofilia, ou ainda participar de um newsgroup formado por uma minoria perseguida sem ter de revelar sua identidade. É claro que existem vários outros exemplos.

Vamos considerar um exemplo específico. Na década de 1990, alguns críticos de um grupo religioso não tradicional postaram suas opiniões em um newsgroup da USENET por meio de um **repostador anônimo**. Esse servidor permitiu que os usuários criassem pseudônimos e enviassem mensagens de correio eletrônico ao servidor, que então reencaminhava ou repostava as mensagens usando o pseudônimo; assim, ninguém poderia saber de onde a mensagem veio de fato.

Algumas postagens revelavam que as afirmações do grupo religioso eram segredos comerciais e documentos protegidos por direitos autorais. O grupo religioso respondeu informando às autoridades locais que seus segredos comerciais tinham sido descobertos e seus direitos autorais infringidos, o que é crime no local onde o servidor se encontrava. Seguiu-se um processo criminal e o operador do servidor foi compelido a entregar às autoridades as informações de mapeamento que revelavam as verdadeiras identidades das pessoas que tinham feito as postagens. (A propósito, essa não foi a primeira vez que um grupo religio-

so ficou insatisfeito porque alguém divulgou seus segredos: William Tyndale foi queimado na fogueira em 1536 por traduzir a Bíblia para o idioma inglês).

Um segmento significativo da comunidade da Internet foi ultrajado por essa brecha de confidencialidade. Todos chegaram à conclusão de que um repostador anônimo que armazena um mapeamento entre endereços reais de correio eletrônico e pseudônimos (chamado repostador do tipo 1) não vale a pena. Esse caso estimulou diversas pessoas a criar repostadores anônimos capazes de resistir a ataques por intimação.

Esses novos repostadores, com frequência chamados **repostadores cypherpunk**, funcionam da maneira ilustrada a seguir. O usuário produz uma mensagem de correio eletrônico completa, com cabeçalhos RFC 822 (exceto *From:*, é claro), codifica a mensagem com a chave pública do repostador e a envia ao repostador. Lá, os cabeçalhos RFC 822 externos são extraídos, o conteúdo é decodificado e a mensagem é repostada. O repostador não tem contas e não mantém nenhum log; assim, mesmo que o servidor seja confiscado mais tarde, não conservará nenhum traço de mensagens que tenham passado por ele.

Muitos usuários que desejam anonimato encadeiam suas solicitações por vários repostadores anônimos, como mostra a Figura 8.48. Aqui, Alice deseja enviar a Bob um cartão pelo dia dos namorados (um cartão realmente anônimo) e para isso utiliza três repostadores. Ela redige a mensagem, *M*, e insere um cabeçalho contendo endereço de correio eletrônico de Bob. Em seguida, codifica toda a mensagem com a chave pública do repostador 3, E_3 (indicada na figura pela hachura horizontal). Para tanto, ela anexa um cabeçalho com o endereço de correio eletrônico do repostador 3 em texto simples. Essa é a mensagem mostrada entre os repostadores 2 e 3 na figura.

Depois, Alice codifica a mensagem com a chave pública do repostador 2, E_2 (indicada pela hachura vertical) e acrescenta um cabeçalho de texto simples contendo o endereço de correio eletrônico do repostador 2. Essa mensagem é mostrada entre 1 e 2 na Figura 8.48. Por fim, ela codifica a mensagem inteira com a chave pública do repostador 1, E_1, e acrescenta um cabeçalho de texto simples com o endereço de correio eletrônico do repostador 1. Essa é a mensagem mostrada à direita de Alice na figura e é a mensagem que ela de fato transmite.

Quando a mensagem chega ao repostador 1, o cabeçalho exterior é extraído. O corpo é decodificado e depois enviado por correio eletrônico para o repostador 2. Etapas semelhantes ocorrem nos outros dois repostadores.

Embora seja extremamente difícil para alguém rastrear a mensagem final de volta até Alice, muitos repostadores tomam precauções adicionais de segurança. Por exemplo, eles podem reter as mensagens por um período de tempo aleatório, adicionar ou remover lixo no fim de uma mensagem e ainda reordenar as mensagens, tudo para tornar mais difícil alguém descobrir que saída de mensagem de um repostador corresponde a cada entrada, a fim de frustrar a análise de tráfego. Para ver a descrição de um sistema que representa o estado da arte em correio eletrônico anônimo, consulte Mazières e Kashoek (1998).

O anonimato não se restringe ao correio eletrônico. Também existem serviços que permitem a navegação anônima na Web usando a mesma forma de caminho em camadas, em que um nó só conhece o próximo nó na cadeia. Esse método é chamado **roteamento cebola**, pois cada nó descasca outra camada para determinar para onde encaminhar o pacote seguinte. O usuário configura seu navegador para usar o serviço anonymizer como um proxy. Tor é um exemplo bem conhecido de tal sistema (Dingledine et al., 2004). Daí em diante, todas as solicitações HTTP vão para o anonymizer, que solicita a página e a devolve. O site vê um nó de saída do anonymizer como a origem da solicitação, não o usuário. Tendo em vista que a rede do anonymizer se recusa a manter um log, depois do fato ninguém pode determinar quem solicitou cada página.

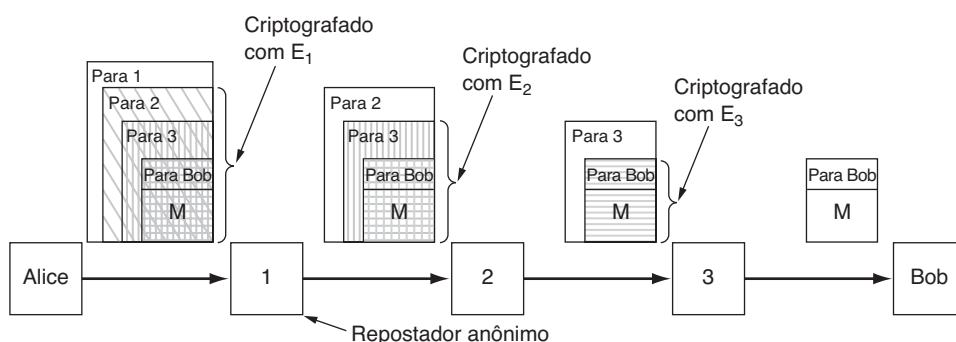



Figura 8.48 | Como Alice utiliza 3 repostadores para enviar uma mensagem a Bob.

8.10.2 Liberdade de expressão

A privacidade se relaciona a indivíduos que desejam restringir o que outras pessoas podem ver sobre eles. Uma segunda questão social importante é a liberdade de expressão e sua oponente, a censura, relacionada com o fato de órgãos governamentais desejarem restringir o que os indivíduos podem ler e publicar. Contendo milhões e milhões de páginas, a Web se tornou um paraíso para o censor. Dependendo da natureza e da ideologia do regime, o material proibido pode incluir sites que contêm quaisquer dos seguintes itens:

1. material impróprio para crianças ou adolescentes;
2. ódio que tem como alvos vários grupos étnicos, religiosos, sexuais ou outros;
3. informações sobre democracia e valores democráticos;
4. relatos de eventos históricos que contradizem a versão do governo;
5. manuais de arrombamento, montagem de armas, codificação de mensagens etc.

A justificativa habitual é proibir os sites considerados 'maus'.

Às vezes, os resultados são inesperados. Por exemplo, algumas bibliotecas públicas instalaram filtros da Web em seus computadores, a fim de torná-los amigáveis para as crianças, bloqueando sites de pornografia. Os filtros vetam os sites contidos em suas listas negras, mas também examinam páginas em busca de palavras obscenas antes de exibi-las. No condado de Loudoun, estado da Virgínia, o filtro bloqueou a busca de informações sobre câncer de mama, porque o filtro encontrou a palavra 'mama'. O patrono da biblioteca processou o condado de Loudoun. Porém, em Livermore, Califórnia, um pai processou a biblioteca pública por *não* instalar um filtro depois que seu filho de 12 anos foi pego vendo um site de pornografia. O que uma biblioteca pode fazer?

Muitas pessoas não percebem que a World Wide Web é de fato uma teia mundial e, portanto, abrange o mundo inteiro. Nem todos os países concordam sobre o que deve ser permitido na Web. Por exemplo, em novembro de 2000, um tribunal francês pediu ao Yahoo! — uma corporação da Califórnia — para impedir que usuários franceses visualizassem leilões de lembranças nazistas no site porque a posse de tal material infringe a lei francesa. O Yahoo apelou a um tribunal dos Estados Unidos que o apoiou, mas a questão das leis que se aplicam a cada país está longe de ser definida.

Imagine estas situações: o que aconteceria se algum tribunal em Utah instruísse a França a bloquear sites que lidassem com vinhos, porque eles não concordam com as leis rigorosíssimas do estado de Utah sobre bebidas alcoólicas? E se a China exigisse que todos os sites sobre democracia fossem proibidos por não ser de interesse do Estado? As leis iranianas sobre religião se aplicam à liberal Suécia? A Arábia Saudita pode bloquear sites que tratam dos direitos das mulheres? A questão inteira é uma verdadeira caixa de Pandora.

Um comentário relevante de John Gilmore é: "A rede interpreta a censura como algo danoso e procura contorná-la". Para ver uma implementação concreta, considere o **serviço eternity** (Anderson, 1996). Seu objetivo é assegurar que informações publicadas não poderão ser retiradas de circulação ou reescritas, como era comum na União Soviética durante o regime de Josef Stalin. Para usar o serviço eternity, o usuário especifica quanto tempo o material deve ser preservado, paga uma taxa proporcional à sua duração e ao tamanho, e o transfere por upload. Daí em diante, ninguém poderá removê-lo ou editá-lo, nem mesmo o próprio usuário que fez o upload.

Como tal serviço poderia ser implementado? O modelo mais simples é usar um sistema não hierárquico, no qual os documentos armazenados seriam colocados em dezenas de servidores participantes, cada um dos quais receberia uma fração da tarifa, e portanto um incentivo para se unir ao sistema. Os servidores devem estar espalhados por muitas jurisdições legais, a fim de proporcionar a máxima resiliência. Listas de dez servidores selecionados ao acaso seriam armazenadas com segurança em vários lugares; assim, se alguma delas fosse comprometida, ainda existiriam outras. Uma autoridade disposta a destruir o documento nunca poderia ter certeza de haver encontrado todas as cópias. O sistema também poderia ser elaborado para reparação automática: caso algumas cópias fossem destruídas, os sites restantes tentariam encontrar novos repositórios para substituí-las.

O serviço eternity foi a primeira proposta de um sistema resistente à censura. Desde então, outros esquemas foram propostos e, em alguns casos, implementados. Diversas características novas foram acrescentadas, como criptografia, anonimato e tolerância a falhas. Com frequência, os arquivos a ser armazenados são divididos em vários fragmentos, com cada fragmento armazenado em muitos servidores. Alguns desses sistemas são o Freenet (Clarke et al., 2002), PASIS (Wylie et al., 2000) e Publius (Waldman et al., 2000). Outro trabalho é relatado em Serjantov (2002).

Muitos países procuram cada vez mais regulamentar a exportação de itens intangíveis, o que em geral inclui sites, software, artigos científicos, correio eletrônico, assistência técnica por telefone e outros. Até mesmo o Reino Unido, com uma tradição de séculos de liberdade de expressão, agora considera seriamente leis bastante restritivas que, por exemplo, definiriam discussões técnicas entre um professor britânico e seu aluno estrangeiro na University of Cambridge como uma forma de exportação regulamentada que necessita de uma licença governamental (Anderson, 2002). É desnecessário dizer que tal política é absurda.

Esteganografia

Em países onde a censura é abundante, os dissidentes com frequência tentam usar a tecnologia para burlar sua rigidez. A criptografia permite que mensagens secretas sejam enviadas (embora talvez isso não seja legal); porém, se o governo imaginar que Alice é uma pessoa nociva, o mero fato de ela se comunicar com Bob pode incluí-lo nessa categoria, pois governos repressivos entendem o conceito de fechamento transitivo, ainda que não tenham muitos matemáticos entre eles. Os repostadores anônimos podem ajudar, mas, se forem proibidos internamente e as mensagens para o exterior exigirem uma licença de exportação do governo, eles não serão muito úteis. No entanto, a Web pode ser de grande auxílio.

Com frequência, as pessoas que desejam se comunicar secretamente tendem a ocultar o fato de haver qualquer comunicação. A ciência de ocultar mensagens é chamada **esteganografia**, das palavras gregas que correspondem a 'escrita cifrada'. Na verdade, os próprios gregos antigos a utilizavam. Heródoto escreveu sobre um general que raspou a cabeça de um mensageiro, tatuou uma mensagem em seu couro cabeludo e deixou o cabelo crescer de novo antes de enviá-lo ao destino. As técnicas modernas são conceitualmente as mesmas, apenas com uma largura de banda mais alta e uma latência mais baixa (e não exigem os serviços de um barbeiro).

Um caso interessante é o da Figura 8.49(a). Essa fotografia, tirada pelo autor no Quênia, contem três zebras contemplando uma acácia. A Figura 8.49(b) parece ter exatamente as mesmas três zebras e a acácia, mas, além disso, ela tem uma atração a mais. A segunda fotografia contém o texto completo de cinco peças de Shakespeare incorporado: *Hamlet*, *Rei Lear*, *Macbeth*, *O mercador de Veneza* e *Julio César*. Juntas, essas peças totalizam mais de 700 KB de texto.

Como funciona esse canal esteganográfico? A imagem em cores original tem 1024 × 768 pixels. Cada pixel consiste em três números de 8 bits, cada um representando a intensidade de uma das cores, vermelha, verde e azul, desse pixel. A cor do pixel é formada pela superposição linear das três cores. O método de codificação esteganográfico utiliza o bit de baixa ordem de cada valor de cor RGB como um canal oculto. Desse modo, cada pixel tem espaço para 3 bits de informações secretas, um no valor vermelho, um no valor verde e um no valor azul. Com uma imagem desse tamanho, podem ser armazenados até 1024 × 768 × 3 bits, ou 294.912 bytes de informações secretas.

O texto completo das cinco peças e uma pequena nota chegam a 734.891 bytes. Primeiro, o texto foi compactado para cerca de 274 KB com um algoritmo de compactação-padrão. A saída compactada foi então criptografada com o uso do IDEA e inserida nos bits de baixa ordem de cada valor de cor. Como podemos ver (ou melhor, como não podemos ver), a existência das informações é completamente invisível. Elas são igualmente invisíveis na versão ampliada e em cores da fotografia. O olho não consegue distinguir com facilidade entre cores de 21 bits e cores de 24 bits.

Para usar a esteganografia na comunicação não detectada, os dissidentes poderiam criar um site repleto de imagens politicamente corretas, como fotografias do Grande Líder, esportes locais, filmes e estrelas de televisão etc. É claro que as figuras estariam recheadas de mensagens esteganográficas. Se as mensagens fossem primeiro compactadas e depois criptografadas, mesmo que alguém suspeitasse de sua presença, teria imensa dificuldade para distinguir as mensagens de ruído branco. É lógico que as imagens devem ser novas; copiar uma figura da Internet e alterar alguns bits é um segredo inútil.

As imagens não são de forma alguma o único tipo de suporte para mensagens esteganográficas. Informações ocultas podem ser transportadas em uma chamada de Voice over IP manipulando os atrasos de pacote, distorcendo o áudio ou

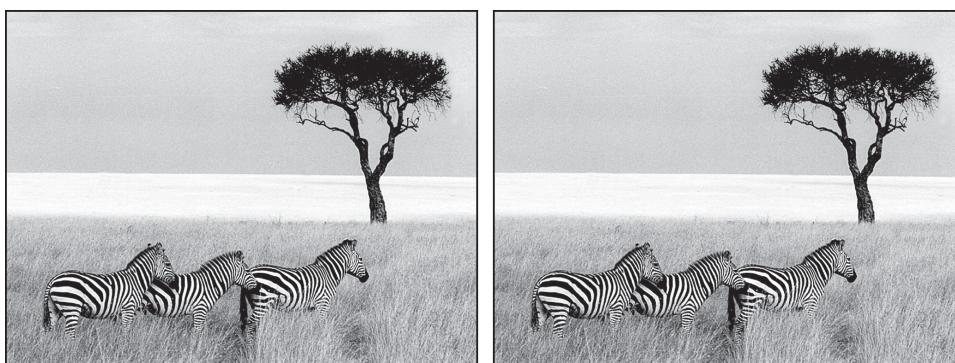



Figura 8.49 | (a) Três zebras e uma árvore. (b) Três zebras, uma árvore e o texto completo de cinco peças de William Shakespeare.

ainda nos campos de cabeçalho dos pacotes (Lubacz et al., 2010). Até mesmo o layout e a ordenação de tags em um arquivo HTML podem transportar informações.

Embora tenhamos examinado a esteganografia no contexto da liberdade de expressão, ela tem vários outros usos. Um uso comum permite que os proprietários de imagens codifiquem mensagens secretas nessas imagens, declarando seus direitos de propriedade. Se tal imagem for roubada e colocada em um site, o dono legal poderá revelar a mensagem esteganográfica no tribunal para provar a quem pertence. Essa técnica é conhecida como **marca d'água**. Ela é descrita em Piva et al. (2002).

Para obter mais informações sobre a esteganografia, consulte Wayner (2008).

8.10.3 Direitos autorais

A privacidade e a censura são apenas duas áreas nas quais a tecnologia encontra a política pública. O **direito autoral** significa a concessão aos criadores de **IP (Intellectual Property)**, incluindo escritores, artistas, compositores, músicos, fotógrafos, cineastas, coreógrafos e outros, do direito exclusivo de explorar sua IP por certo período de tempo, em geral durante a vida do autor somada a 50 anos ou 75 anos, no caso da propriedade corporativa. Depois de expirar o período de proteção pelos direitos autorais de uma obra, ela passa para o domínio público e qualquer pessoa pode usá-la ou vendê-la como desejar. Por exemplo, o Projeto Gutenberg (www.promo.net/pg) colocou milhares de obras de domínio público (por exemplo, obras de Shakespeare, Twain, Dickens) na Web. Em 1998, a pedido de Hollywood, o Congresso dos Estados Unidos estendeu os direitos autorais nos EUA por mais 20 anos. O pessoal do cinema alegava que, sem uma extensão desse período, ninguém criaria mais nada. Em contraste, as patentes duram apenas vinte anos e, mesmo assim, as pessoas não param de apresentar novas invenções.

A discussão sobre os direitos autorais ganhou espaço quando o Napster, um serviço de troca de obras musicais, alcançou 50 milhões de membros. Embora o Napster realmente não copiasse nenhuma música, os tribunais sustentaram que manter um banco de dados central de quem tinha uma cópia de cada música era infração contribuinte, isto é, ele ajudava outras pessoas a infringir a lei. Apesar de ninguém afirmar que os direitos autorais sejam má ideia (embora muitos reclamem que o processo é muito longo, favorecendo assim as grandes empresas em detrimento do público), a próxima geração de compartilhamento de música já está levantando questões éticas importantes.

Por exemplo, considere uma rede não hierárquica em que pessoas compartilham arquivos legais (música de domínio público, vídeos domésticos, tratados religiosos que não representem segredos comerciais etc.) e talvez alguns deles sejam protegidos por direitos autorais. Suponha que todas essas pessoas estejam on-line o tempo todo, por meio de ADSL ou cabo. Cada máquina tem um índice do que está no disco rígido, além de uma lista de outros membros. Alguém que procurar um item específico pode escolher um membro ao acaso e ver se ele tem o item. Caso contrário, a pessoa pode procurar o item em todos os membros da lista desse primeiro membro, e depois em todos os membros das listas desses outros membros, e assim por diante. Os computadores são muito bons nesse tipo de trabalho. Tendo encontrado o item, o solicitante simplesmente o copia.

Se o trabalho estiver protegido por direitos autorais, as chances são de que o solicitante esteja infringindo a lei (embora, no caso de transferências internacionais, não esteja claro que lei deve ser aplicada). Entretanto, como classificar o fornecedor? É crime manter em seu disco rígido uma música pela qual você pagou e baixou legalmente, apenas porque outras pessoas podem encontrá-la? Se você tem uma cabana no campo e um ladrão de IP invade sua cabana levando um notebook e um scanner, copia um livro protegido por direitos autorais e escapa sorrateiramente, você é culpado do crime de deixar de proteger os direitos autorais de outra pessoa?

No entanto, existem mais dificuldades nessa área. Há uma grande batalha entre Hollywood e a indústria de informática. Hollywood deseja a proteção rígida de toda a propriedade intelectual, e a indústria de informática não quer ser a polícia a serviço de Hollywood. Em outubro de 1998, o Congresso norte-americano aprovou o **DMCA (Digital Millennium Copyright Act)**, que torna crime frustrar qualquer mecanismo de proteção presente em uma obra protegida por direitos autorais ou informar outras pessoas sobre como lográ-lo. Legislação semelhante está surgindo na União Europeia. Embora quase ninguém pense que piratas do Extremo Oriente devam ter permissão para duplicar obras protegidas, muitas pessoas imaginam que o DMCA desloca completamente o equilíbrio entre o interesse do detentor dos direitos autorais e o interesse público.

Vejamos um exemplo prático. Em setembro de 2000, um consórcio da indústria da música encarregado de elaborar um sistema inviolável para venda de obras musicais on-line patrocinou um concurso convidando pessoas a tentar violar o sistema (exatamente o que deve ser feito no caso de qualquer sistema de segurança novo). Pesquisadores da área de segurança provenientes de várias universidades formaram uma equipe, liderada pelo professor Edward Felten de Princeton, que aceitou o desafio e conseguiu quebrar o sistema. Em seguida, eles escreveram um artigo sobre suas descobertas e o submeteram a uma conferência de segurança do USENIX, em que esse artigo passou por uma revisão e foi aceito. Antes de apresentar seu trabalho, Felten recebeu uma carta da Recording Industry Association of America que ameaçava processar os autores com base no DMCA

se eles publicassem o artigo. A resposta dos pesquisadores foi abrir um processo pedindo que um tribunal federal decidisse se a publicação de documentos científicos sobre pesquisas na área de segurança ainda era legal. Temendo uma decisão definitiva dos tribunais contra ela, a indústria retirou sua ameaça e o tribunal rejeitou a ação de Felten. Sem dúvida, os fabricantes de discos foram motivados pela fragilidade de sua posição: eles haviam convidado pessoas a tentar violar seu sistema, e depois ameaçaram processar algumas delas por aceitar o desafio. Com a ameaça retirada, o ensaio foi publicado (Craver et al., 2001). Uma nova disputa é quase certa.

Enquanto isso, música e filmes pirateados alimentaram o crescimento maciço das redes peer-to-peer. Isso não agradou as mantenedores de direito autoral, que usavam o DMCA para tomar ações. Atualmente, existem sistemas automatizados que procuram redes peer-to-peer e depois disparam avisos para operadores de rede e usuários que são suspeitos de infringir direitos autorais. Nos Estados Unidos, esses avisos são conhecidos como **notas de desativação do DMCA**. Essa busca é uma corrida de braços, pois é difícil apanhar infratores de direitos autorais de modo confiável. Até mesmo seu tipógrafo poderia ser confundido com um criminoso (Piatek et al., 2008).

Uma questão relacionada a essa é a extensão da **doutrina de uso legal**, estabelecida por decisões judiciais em vários países. Essa doutrina afirma que os compradores de uma obra protegida por direitos autorais têm certos direitos limitados de copiar a obra, inclusive o direito de citar partes dela para fins científicos, usá-la como material didático em escolas ou faculdades e, em alguns casos, criar cópias de reserva para uso pessoal no caso de falha do meio original. Os testes para definir o que constitui uso legal incluem (1) se o uso é comercial, (2) que porcentagem do todo está sendo copiada e (3) o efeito da cópia sobre as vendas da obra. Tendo em vista que o DMCA e leis semelhantes dentro da União Europeia proíbem frustrar os esquemas de proteção contra cópia, essas leis também proíbem o uso legal. Na realidade, o DMCA prejudica os direitos históricos dos usuários para dar mais poder aos vendedores de conteúdo. É inevitável um confronto.

Outro desenvolvimento na área que reduz a importância até mesmo do DMCA em seu deslocamento do equilíbrio entre os detentores de direitos autorais e os usuários é a **computação confiável**, defendida por setores da indústria como o **TCG** (**Trusted Computing Group**), liderado por empresas como Intel e Microsoft. A ideia é oferecer suporte para monitoramento cuidadoso do comportamento do usuário em diversos aspectos (por exemplo, reprodução de música pirateada) em um nível abaixo do sistema operacional, a fim de proibir o comportamento indesejável. O sistema permite que o software escrito por detentores de conteúdo manipule PCs de maneiras que os usuários não possam mudar. Isso levanta a questão de quem é confiável na confiável computação. Certamente, não é o usuário. É desnecessário dizer que as consequências sociais desse esquema são imensas. É ótimo que a indústria esteja finalmente prestando atenção à segurança, mas é lamentável que ela esteja inteiramente voltada para impor a lei de direitos autorais, em vez de lidar com vírus, crackers, intrusos e outras questões de segurança com as quais a maioria das pessoas está preocupada.

Em resumo, os legisladores e juristas estarão ocupados tentando equilibrar os interesses econômicos dos proprietários de direitos autorais com o interesse público nos próximos anos. O espaço virtual não é diferente do espaço físico: ele constantemente joga um grupo contra outro, resultando em lutas pelo poder, litígio e (esperamos) por fim algum tipo de resolução, pelo menos até surgir alguma nova tecnologia capaz de resolver a situação.

8.11 Resumo

A criptografia é uma ferramenta que pode ser usada para manter informações confidenciais e garantir sua integridade e autenticidade. Todos os sistemas criptográficos modernos se baseiam no principio de Kerckhoff de um algoritmo publicamente conhecido e uma chave secreta. Muitos algoritmos criptográficos usam transformações complexas que envolvem substituições e permutações para transformar o texto simples em texto cifrado. Porém, se a criptografia quântica puder se tornar prática, o uso de blocos de uma única vez poderá fornecer sistemas criptográficos verdadeiramente invioláveis.

Os algoritmos criptográficos podem ser divididos como de chave simétrica e de chave pública. Os algoritmos de chave simétrica desfiguram os bits em uma série de rodadas parametrizados pela chave para transformar o texto simples no texto cifrado. AES (Rijndael) e o DES triplo são os algoritmos de chave simétrica mais populares no momento. Esses algoritmos podem ser usados no modo livro de códigos eletrônicos, modo blocos de cifras encadeados, modo fluxo de cifra, modo contador e outros.

Nos algoritmos de chave pública, são usadas chaves diferentes para codificação e decodificação, e a chave de decodificação não pode ser derivada a partir da chave de codificação. Essas propriedades tornam possível divulgar a chave pública. O principal algoritmo de chave pública é o RSA, cuja força deriva da grande dificuldade de fatorar números muito grandes.

Documentos legais, comerciais e outros precisam ser assinados. Consequentemente, foram criados vários esquemas de assinaturas digitais, empregando algoritmos de chave simétrica e algoritmos de chave pública. Em geral, as mensagens que devem ser assinadas são submetidas a um hash com a utilização de algoritmos como SHA-1, e então o hash é assinado em lugar das mensagens originais.

O gerenciamento de chaves públicas pode ser implementado com o emprego de certificados, documentos que

vinculam um protagonista a uma chave pública. Os certificados são assinados por uma autoridade confiável ou por alguém aprovado (recursivamente) por uma autoridade confiável. A raiz da cadeia tem de ser obtida com antecedência, mas os navegadores em geral têm muitos certificados de raiz embutidos.

Essas ferramentas criptográficas podem ser usadas para proteger o tráfego de rede. O IPsec opera na camada de rede, codificando fluxos de pacotes de host para host. Os firewalls podem efetuar a triagem do tráfego que entra ou sai de uma organização, muitas vezes com base no protocolo e na porta utilizados. As redes privadas virtuais podem simular uma antiga rede dedicada para oferecer certas propriedades de segurança desejáveis. Por fim, as redes sem fios precisam de uma boa segurança a fim de evitar que mais alguém leia todas as mensagens, e protocolos como 802.11i oferecem isso.

Quando duas partes estabelecem uma sessão, elas têm de autenticar uma à outra e, se necessário, estabelecer uma chave de sessão compartilhada. Existem diversos protocolos de autenticação, incluindo alguns que usam uma terceira parte confiável, Diffie-Hellman, Kerberos e criptografia de chave pública.

A segurança de correio eletrônico pode ser alcançada por uma combinação das técnicas que estudamos neste capítulo. Por exemplo, o PGP compacta as mensagens, depois as codifica usando uma chave secreta e envia essa chave codificada com a chave pública do receptor. Além disso, ele também efetua o hash da mensagem e envia o hash assinado para confirmar a integridade da mensagem.

A segurança da Web também é um tópico importante, começando com a nomenclatura segura. O DNSsec oferece um modo de evitar o DNS spoofing, bem como realizar a certificação automática de nomes. A maioria dos sites de comércio eletrônico na Web utiliza SSL/TLS para estabelecer sessões autenticadas e seguras entre o cliente e o servidor. São usadas várias técnicas para lidar com código móvel, em especial caixas de brita e assinatura de código.

A Internet levanta muitas questões em que a tecnologia interage fortemente com a política pública. Algumas áreas relevantes incluem privacidade, liberdade de expressão e direitos autorais.

Problemas

1. Resolva a cifra monoalfabética a seguir. O texto simples, formado apenas por letras, é um trecho de um conhecido poema de Lewis Carroll.

   mvyy bek mnyx n yvjjyr snijrh invq n muvjvdt je n idnvy

   jurhri n fehfevir pyeir oruvdq ki ndq uri jhrnqvdt ed zb jnvy

   Irr uem rntrhyb jur yeoijrhi ndq jur jkhjyri nyy nqlndpr

   Jurb nhr mnvjvdt ed jur iuvdtyr mvyy bek pezr ndq wevd jur qndpr

   mvyy bek, medj bek, mvyy bek, medj bek, mvyy bek wevd jur qndpr

   mvyy bek, medj bek, mvyy bek, medj bek, medj bek wevd jur qndpr

2. Uma cifra afim é uma versão de uma cifra de substituição monoalfabética, em que as letras de um alfabeto de tamanho m são primeiro mapeadas para os inteiros no intervalo de 0 a $m - 1$. Na sequência, o inteiro representando cada letra em texto simples é transformado em um inteiro representando a letra correspondente no texto. A função de codificação para uma única letra é $E(x) = (ax + b) \bmod m$, onde m é o tamanho do alfabeto e a e b são a chave da cifra, e são primos entre si. Trudy descobre que Bob gerou um texto cifrado usando uma cifra afim. Ela apanha uma cópia do texto cifrado e descobre que a letra mais frequente do texto cifrado é 'R', e a segunda letra mais frequente do texto cifrado é 'K'. Mostre como Trudy pode quebrar o código e recuperar o texto simples.

3. Resolva a seguinte cifra de transposição de colunas. O texto simples foi extraído de um livro sobre computadores; portanto, 'computer' é uma palavra muito provável. O texto simples é formado apenas por letras (sem espaços). O texto cifrado está dividido em blocos de cinco caracteres para proporcionar melhor legibilidade.

   aauan cvlre rurnn dltme aeepb ytust iceat npmey iicgo gorch srsoc

   nntii imiha oofpa gsivt tpsit lbolr otoex

4. Alice usou uma cifra de transposição para codificar suas mensagens para Bob. Para aumentar a segurança, ela criptografou a chave da cifra de transposição usando uma cifra de substituição e manteve a cifra criptografada em seu computador. Trudy conseguiu se apossar da chave da cifra de transposição criptografada. Poderá ela decifrar as mensagens de Alice para Bob? Por quê?

5. Encontre um bloco de 77 bits que gere o texto 'Hello World' a partir do texto cifrado da Figura 8.3.

6. Você é um espião e, de modo conveniente, tem uma biblioteca com um número infinito de livros à sua disposição. Seu operador também tem essa biblioteca à disposição. Você concordou em usar o *Senhor dos Anéis* como uma chave única. Explique como você poderia usar esses recursos para gerar uma chave única infinitamente longa.

7. A criptografia quântica exige uma pistola de fótons que possa, por demanda, disparar um único fóton transportando 1 bit. Neste problema, calcule quantos fótons um bit transporta em um enlace de fibra de 250 Gbps. Suponha que o comprimento de um fóton seja igual a seu comprimento de onda que, para fins deste problema, é 1 micron. A velocidade da luz na fibra é 20 cm/ns.

8. Se Trudy capturar e regenerar fótons quando a criptografia quântica estiver em uso, ela obterá alguns fótons errados e provocará o surgimento de erros na chave única de Bob. Em média, que fração dos bits do bloco de uma só vez de Bob estarão errados?

9. Um princípio criptográfico fundamental estabelece que todas as mensagens devem ter redundância. Porém, também sabemos que a redundância ajuda um intruso a saber se uma chave hipotética está correta. Considere duas formas de redundância. Primeiro, os n bits iniciais do texto simples contêm um padrão conhecido. Segundo, os n bits finais da mensagem contêm um hash sobre a mensagem. Do ponto de vista da segurança, essas duas alternativas são equivalentes? Comente sua resposta.
10. Na Figura 8.5, as caixas P e S se alternam. Apesar da aparência esteticamente agradável dessa organização, não seria mais seguro primeiro ter todas as caixas P e depois todas as caixas S? Comente sua resposta.
11. Projete um ataque ao DES baseado no conhecimento de que o texto simples é formado exclusivamente por letras ASCII em caixa alta, além de espaço, vírgula, ponto, ponto e vírgula, retorno de cursor e avanço de linha. Nada é conhecido sobre os bits de paridade do texto simples.
12. No texto, calculamos que uma máquina de análise de cifras com um milhão de processadores que pudesse analisar uma chave em 1 nanossegundo demoraria apenas 10^{16} anos para quebrar a versão de 128 bits do AES. Vamos calcular quanto tempo levará desta vez para reduzir para 1 ano, o que, logicamente, ainda é muito tempo. Para conseguir isso, precisamos que os computadores sejam 10^{16} vezes mais rápidos. Se a lei de Moore (a potência de computação duplica a cada 18 meses) continuar a ser válida, quantos anos serão necessários até que um computador paralelo possa reduzir o tempo de quebra da senha para um ano?
13. O AES admite uma chave de 256 bits. Quantas chaves tem o AES-256? Veja se é possível encontrar algum número em física, química ou astronomia com aproximadamente o mesmo tamanho. Use a Internet para ajudá-lo a procurar números grandes. Tire uma conclusão de sua pesquisa.
14. Suponha que uma mensagem tenha sido criptografada com a utilização do DES no modo de contador. Um bit do texto cifrado no bloco C_i é acidentalmente transformado de 0 para 1 durante a transmissão. Quanto do texto simples será adulterado em decorrência disso?
15. Agora, considere o encadeamento de blocos de texto cifrado. Em vez de um único bit 0 ser transformado em um bit 1, um bit 0 extra é inserido no fluxo de texto cifrado depois do bloco C_i. Que proporção do texto simples será adulterado em decorrência disso?
16. Compare o encadeamento de blocos de cifras com o modo de feedback de cifra no que se refere ao número de operações necessárias para a transmissão de um arquivo muito grande. Qual dos dois é o mais eficiente e em que proporção?
17. Usando o sistema de criptografia de chave pública RSA, com $a = 1$, $b = 2$ ... $y = 25$, $z = 26$.
    (a) Se $p = 5$ e $q = 13$, cite cinco valores válidos para d.
    (b) Se $p = 7$ e $q = 11$, cite cinco valores válidos para d.
    (c) Usando $p = 5$, $q = 11$ e $d = 9$, encontre e e criptografe “hello”.
18. Alice e Bob usam a criptografia de chave privada RSA para se comunicarem. Trudy descobre que Alice e Bob compartilharam um dos primos usados para determinar o número n de seus pares de chave pública. Em outras palavras, Trudy descobriu que $n_a = p_a \times q$ e $n_b = p_b \times q$. Como Trudy pode usar essa informação para quebrar o código de Alice?
19. Considere o uso do modo de contador, como mostra a Figura 8.13, mas com $IV = 0$. O uso de 0 ameaça a segurança da cifra em geral?
20. Na Figura 8.17, vemos como Alice pode enviar a Bob uma mensagem assinada. Se Trudy substituir P, Bob poderá descobrir. No entanto, o que acontecerá se Trudy substituir ao mesmo tempo P e a assinatura?
21. As assinaturas digitais têm uma deficiência potencial devido a usuários preguiçosos. Em transações de comércio eletrônico, um contrato poderia ser interrompido e o usuário poderia ser solicitado a assinar seu hash SHA-1. Se o usuário não verificar realmente que o contrato e o hash correspondem, ele poderá assinar inadvertidamente um contrato diferente. Suponha que a Máfia tente explorar essa fraqueza para ganhar algum dinheiro. Os mafiosos configuram um site com pagamento (por exemplo, pornografia, jogo etc.) e pedem aos novos clientes um número de cartão de crédito. Em seguida, enviam um contrato ao cliente confirmando que este deseja usar seus serviços e que pagará com o cartão de crédito; depois, solicitam que o cliente assine o contrato, sabendo que a maioria dos clientes simplesmente assinará sem verificar se o contrato e o hash coincidem. Mostre como a Máfia pode comprar diamantes pela Internet de um joalheiro legítimo e debitá-los de clientes desatentos.
22. Uma turma de matemática tem 20 alunos. Supondo que todos os nasceram no primeiro semestre do ano — entre 1 de janeiro e 30 de junho —, qual é a probabilidade de pelo menos dois alunos fazerem aniversário no mesmo dia? Suponha que ninguém tenha nascido em um ano bissexto, de modo que existem 181 dias possíveis.
23. Depois de Ellen ter confessado a Marilyn que a enganou no episódio da indicação de Tom, Marilyn resolveu evitar esse problema ditando o conteúdo de futuras mensagens a um gravador e fazendo sua nova secretária simplesmente digitá-las. Marilyn planejava examinar as mensagens em seu terminal depois de ser digitadas, para ter certeza de que continham suas palavras exatas. A nova secretária ainda pode usar o ataque do aniversário para falsificar uma mensagem? De que maneira? *Dica*: Ela pode fazê-lo.
24. Considere a tentativa malsucedida de Alice de conseguir a chave pública de Bob na Figura 8.20. Suponha que Bob e Alice já compartilhem uma chave secreta, mas Alice ainda quer a chave pública de Bob. Agora existe um modo de obtê-la em segurança? Em caso afirmativo, como?
25. Alice quer se comunicar com Bob, usando a criptografia de chave pública. Ela estabelece uma conexão com alguém que espera que seja Bob. Alice pede a ele sua

chave pública e ele a envia em texto simples, juntamente com um certificado X.509 assinado pela CA raiz. Alice já tem a chave pública da CA raiz. Que etapas ela deve executar para verificar se está se comunicando com Bob? Suponha que Bob não se importe em saber com quem está se comunicando (por exemplo, Bob e alguma espécie de serviço público).

26. Suponha que um sistema utilize a PKI baseada em uma hierarquia estruturada em árvore de CAs. Alice quer se comunicar com Bob e recebe um certificado de Bob assinado por uma CA *X*, depois de estabelecer um canal de comunicação com Bob. Suponha que Alice nunca tenha ouvido falar em *X*. Que etapas Alice deve executar para confirmar que está se comunicando com Bob?
27. O IPsec usando a AH pode ser empregado em modo de transporte quando uma das máquinas está atrás de um NAT? Explique sua resposta.
28. Alice deseja enviar uma mensagem a Bob usando hashes SHA-1. Ela consulta você em relação ao algoritmo de assinatura apropriado que será usado. O que você sugeriria?
29. Apresente uma razão que justifique o fato de um firewall poder ser configurado de modo a inspecionar o tráfego de entrada. Apresente uma razão que justifique o fato de um firewall poder ser configurado de modo a inspecionar o tráfego de saída. Você acredita que as inspeções têm probabilidade de sucesso?
30. Suponha que uma organização utilize uma VPN para se conectar em segurança a seus sites pela Internet. Jim, um usuário na organização, usa a VPN para se comunicar com sua chefe Mary. Descreva um tipo de comunicação entre Jim e Mary que não exigiria o uso de criptografia ou outro mecanismo de segurança, e outro tipo de comunicação que exigiria criptografia ou outros mecanismos de segurança. Explique sua resposta.
31. Faça uma pequena alteração em uma mensagem no protocolo da Figura 8.30, de modo a torná-lo resistente ao ataque por reflexão. Explique por que sua mudança funciona.
32. A troca de chaves de Diffie-Hellman está sendo usada para estabelecer uma chave secreta entre Alice e Bob. Alice envia a Bob a mensagem (227, 5, 82). Bob responde com (125). O número secreto de Alice, *x*, é 12, e o número secreto de Bob, *y*, é 3. Mostre como Alice e Bob calculam a chave secreta.
33. Dois usuários podem estabelecer uma chave secreta compartilhada usando o algoritmo de Diffie-Hellman, mesmo que nunca tenham se encontrado, não compartilhem segredos e não tenham certificados.
    (a) Explique como esse algoritmo é suscetível a um ataque do homem no meio.
    (b) Como essa suscetibilidade mudaria se *n* ou *g* fossem secretos?
34. No protocolo da Figura 8.35, por que *A* é enviado em texto simples junto com a chave de sessão criptografada?
35. No protocolo de Needham-Schroeder, Alice gera dois desafios, R_A e R_{A2}. Isso parece exagero. Apenas um não seria suficiente?
36. Suponha que uma organização utilize o Kerberos para autenticação. Em termos de segurança e disponibilidade de serviço, qual será o efeito se AS ou TGS for desativado?
37. Alice está usando o protocolo de autenticação por chave pública da Figura 8.39 para autenticar a comunicação com Bob. Porém, ao enviar a mensagem 7, Alice se esqueceu de criptografar R_B. Trudy agora conhece o valor de R_B. Alice e Bob precisam repetir o procedimento de autenticação com novos parâmetros a fim de garantir a comunicação segura? Explique sua resposta.
38. No protocolo de autenticação por chave pública da Figura 8.39, na mensagem 7, R_B é criptografado com K_S. Essa criptografia é necessária, ou teria sido melhor enviar a mensagem de volta em texto simples? Explique sua resposta.
39. Terminais de pontos de venda que utilizam cartões com tarja magnética e códigos PIN têm uma falha fatal: um comerciante inescrupuloso pode modificar sua leitora de cartões para armazenar todas as informações do cartão, assim como o código PIN, a fim de informar outras transações (falsas) no futuro. A próxima geração de terminais de pontos de venda utilizará cartões com uma CPU completa, teclado e um pequeno visor. Imagine um protocolo para esse sistema que comerciantes inescrupulosos não consigam burlar.
40. Seria possível enviar uma mensagem PGP por multicast? Que restrições se aplicariam?
41. Supondo que todos na Internet usassem o PGP, uma mensagem PGP poderia ser enviada a um endereço qualquer da Internet e ser decodificada corretamente por todos os envolvidos? Comente sua resposta.
42. O ataque mostrado na Figura 8.42 omite uma etapa. A etapa não é necessária para o spoofing funcionar, mas incluí-la poderia reduzir a suspeita potencial depois do fato. Qual é a etapa omitida?
43. O protocolo de transporte de dados SSL envolve dois nonces, bem como uma chave pré-mestre. Que valor, se for o caso, tem o uso dos nonces?
44. Considere uma imagem de 2048 × 512 pixels. Você deseja criptografar um arquivo com tamanho de 2,5 MB. Que fração do arquivo você pode criptografar nessa imagem? Que fração você poderia criptografar se compactasse o arquivo para um quarto de seu tamanho original? Mostre seus cálculos.
45. A imagem da Figura 8.49 (b) contém o texto ASCII de cinco peças de Shakespeare. Seria possível ocultar música em vez de texto entre as zebras? Em caso afirmativo, como isso seria feito e que quantidade de música você ocultaria na imagem? Em caso negativo, por que não?
46. Você recebe um arquivo de texto de 60 MB, que deve ser criptografado usando a esteganografia nos bits de baixa ordem de cada cor em um arquivo de imagem. Que tamanho de imagem seria necessário para poder criptografar o arquivo inteiro? Que tamanho seria necessário se o arquivo fosse primeiro compactado para um terço de seu tamanho original? Mostre suas respostas em pixels e apresente os cálculos. Suponha que as imagens tenham uma relação entre os eixos de 3:2, por exemplo, 3000 x2000 pixels.

47. Alice era uma usuária pesada de um repostador anônimo do tipo 1. Ela postava muitas mensagens em seu newsgroup favorito, alt.fanclub.alice, e todo mundo sabia que as mensagens eram todas de Alice, porque tinham o mesmo pseudônimo. Supondo-se que o repostador funcionasse corretamente, Trudy não conseguiria se fazer passar por Alice. Depois que os repostadores do tipo 1 foram desativados, Alice trocou para um repostador cypherpunk e iniciou um novo tópico thread de mensagens em seu newsgroup. Elabore um meio de impedir que Trudy possa postar novas mensagens para o newsgroup, fazendo-se passar por Alice.
48. Procure na Internet algum caso interessante envolvendo privacidade e escreva um relatório de uma página sobre o tema.
49. Procure na Internet algum caso jurídico envolvendo direitos autorais (*copyright*) *versus* uso legal (*fair use*) e escreva um relatório de uma página resumindo os resultados de sua pesquisa.
50. Escreva um programa que codifique sua entrada por meio de uma operação XOR entre ela e um fluxo de chaves. Descubra ou desenvolva o melhor gerador de números aleatórios que puder para gerar o fluxo de chaves. O programa deve atuar como um filtro, recebendo texto simples na entrada padrão e gerando texto cifrado na saída padrão (e vice-versa). O programa deve receber um parâmetro, a chave que produz a semente do gerador de números aleatórios.
51. Escreva um procedimento que calcule o hash SHA-1 de um bloco de dados. O procedimento deve ter dois parâmetros: um ponteiro para o buffer de entrada e um ponteiro para um buffer de saída de 20 bytes. Para ver a especificação exata do SHA-1, procure na Internet FIPS 180-1, a especificação completa.
52. Escreva uma função que aceite um fluxo de caracteres ASCII e criptografe essa entrada usando uma cifra por substituição com o modo de encadeamento de blocos de cifras. O tamanho do bloco deverá ser de 8 bytes. O programa deverá apanhar o texto simples da entrada padrão e imprimir o texto cifrado na saída padrão. Para este problema, você poderá selecionar qualquer sistema razoável para determinar o final da entrada, e/ou quando o preenchimento deverá ser aplicado para completar o bloco. Você pode selecionar qualquer formato de saída, desde que não seja ambíguo. O programa deverá receber dois parâmetros:
    1. Um ponteiro para o vetor de inicialização; e
    2. Um número k, representando o deslocamento da cifra de substituição, de modo que cada caractere ASCII seja codificado pelo k-ésimo caractere adiante dele no alfabeto.

    Por exemplo, se $x = 3$, então A é codificado por D, B é codificado por E etc. Faça suposições razoáveis com relação a alcançar o último caractere no conjunto ASCII. Documente claramente em seu código quaisquer suposições que você faça sobre a entrada e o algoritmo de criptografia.
53. A finalidade deste problema é dar-lhe um melhor conhecimento sobre os mecanismos do RSA. Escreva uma função que receba como parâmetros números primos p e q, calcule chaves RSA públicas e privadas usando esses parâmetros e escreva n, z, d e e na saída-padrão. A função também deverá aceitar um fluxo de caracteres ASCII e criptografar essa entrada usando as chaves RSA calculadas. O programa deverá apanhar o texto simples da entrada-padrão e imprimir o texto cifrado na saída-padrão. A criptografia deverá ser executada um caractere por vez, ou seja, apanhando cada caractere na entrada e criptografando-o independentemente dos outros. Para este problema, você poderá selecionar qualquer sistema razoável para determinar o final da entrada. Você pode selecionar qualquer formato de saída, desde que não seja ambíguo. Documente claramente em seu código quaisquer suposições que você faça sobre a entrada e o algoritmo de criptografia.

Sugestões de leitura e referências bibliográficas

Capítulo

Terminamos nosso estudo das redes de computadores, mas isso é apenas o começo. Muitos tópicos interessantes não foram tratados com tantos detalhes quanto merecem, e outros foram omitidos completamente por falta de espaço. Neste capítulo, oferecemos algumas sugestões de leitura e uma extensa bibliografia na área, para benefício dos leitores que desejam continuar seus estudos em redes de computadores.

9.1 Sugestões de leitura

Há muita leitura sobre todos os aspectos das redes de computadores. Dois periódicos que publicam artigos nessa área são *IEEE/ACM Transactions on Networking* e *IEEE Journal on Selected Areas in Communications*.

Os periódicos da ACM Special Interest Groups on Data Communications (SIGCOMM) e Mobility of Systems, Users, Data, and Computing (SIGMO-BILE) publicam muitos artigos de interesse, em especial sobre assuntos emergentes. São eles: *Computer Communication Review* e *Mobile Computing and Communications Review*.

O IEEE também publica três revistas — *IEEE Internet Computing*, *IEEE Network Magazine* e *IEEE Communications Magazine* — que contêm estudos, tutoriais e estudos de caso sobre redes. As duas primeiras enfatizam arquitetura, padrões e software, e a última visa à tecnologia de comunicações (fibra óptica, satélites e assim por diante).

Existem inúmeras conferências anuais ou bianuais que atraem diversos artigos sobre redes. Em particular, procure a conferência *SIGCOMM*, *NSDI* (Symposium on Networked Systems Design and Implementation), *MobiSys* (Conference on Mobile Systems, Applications and Services), *SOSP* (Symposium on Operating Systems Principles) e *OSDI* (Symposium on Operating Systems Design and Implementation).

A seguir, listamos algumas sugestões para leitura complementar, ligadas aos capítulos deste livro. Muitas delas são livros ou capítulos de livros, com alguns tutoriais e estudos. As referências completas estão na Seção 9.2.

9.1.1 Introdução e trabalhos na área

Comer, *The Internet Book*, 4.ed.

Quem quer que procure uma introdução descomplicada à Internet deverá olhar aqui. Comer descreve a história, o crescimento, a tecnologia, os protocolos e os serviços da Internet em termos que os iniciantes conseguem entender, mas há tanto material no livro que ele também é de interesse para os leitores mais técnicos.

Computer Communication Review, 25th Anniversary Issue, jan. 1995

Para um exame em primeira mão de como a Internet se desenvolveu, essa edição especial reúne artigos importantes até 1995. Inclui ensaios que mostram o desenvolvimento do TCP, multicast, o DNS, Ethernet e a arquitetura em geral.

Crovella e Krishnamurthy, *Internet Measurement*

Como podemos avaliar o funcionamento da Internet, de qualquer maneira? Essa pergunta não é fácil de ser respondida, pois não há um responsável pela Internet. O livro descreve as técnicas desenvolvidas para medir a operação, da infraestrutura de rede às aplicações.

IEEE Internet Computing, jan.–fev. 2000

A primeira edição do *IEEE Internet Computing* do milênio fez como você esperaria: pediu às pessoas que ajudaram a criar a Internet que especulassem para onde ela caminharia neste milênio. Os especialistas são Paul Baran, Lawrence Roberts, Leonard Kleinrock, Stephen Crocker, Danny Cohen, Bob Metcalfe, Bill Gates, Bill Joy e outros. Veja se suas previsões se confirmaram uma década depois.

Kipnis, 'Beating the System: Abuses of the Standards Adoption Process'

Comitês de padrões tentam ser imparciais e independentes do fornecedor, mas infelizmente existem empresas que tentam abusar do sistema. Por exemplo, já aconteceu várias vezes de uma empresa ajudar a desenvolver um padrão e, depois de tê-lo aprovado, anunciar que ele se baseia em uma patente de sua propriedade e será licenciado somente para as empresas que ela quiser, a preços por ela determinados. Para ter uma visão da face obscura da padronização, o artigo é um excelente começo.

Hafner e Lyon, *Where Wizards Stay Up Late*

Naughton, *A Brief History of the Future*

Afinal, quem inventou a Internet? Muitas pessoas reivindicaram o crédito. E com razão, pois muita gente participou, de diversas maneiras. Paul Baran escreveu um relatório que descreve a comutação de pacotes; pessoas em diversas universidades projetaram a arquitetura ARPANET;

o pessoal das BBN programou os primeiros IMPS; Bob Kahn e Vint Cerf inventaram o TCP/IP; e assim por diante. Esses livros contam a história da Internet, pelo menos até 2000, repletos de muitas anedotas.

9.1.2 A camada física

Bellamy, *Digital Telephony*, 3.ed.

Para ver um retrospecto da outra rede importante, a de telefonia, esse livro digno de crédito tem tudo o que você sempre quis saber, e mais. Interessantes em especial são os capítulos sobre transmissão e multiplexação, comutação digital, fibra óptica, telefonia móvel e DSL.

Hu e Li, 'Satellite-Based Internet: a Tutorial'

O acesso à Internet via satélite é diferente do uso de linhas terrestres. Não há apenas a questão do atraso, mas o roteamento e a comutação também são diferentes. Nesse artigo, os autores examinam as questões ligadas ao uso de satélites para acesso à Internet.

Joel, 'Telecommunications and the IEEE Communications Society'

Para ver uma história compacta mas abrangente das telecomunicações, que começa com o telégrafo e termina com o 802.11, esse é o artigo a pesquisar. Ele também abrange rádio, telefones, comutação analógica e digital, cabos submarinos, transmissão digital, broadcasting de televisão, satélites, TV a cabo, comunicações ópticas, telefones móveis, comutação de pacotes, a ARPANET e a Internet.

Palais, *Fiber Optic Communication*, 5.ed.

Os livros sobre tecnologia de fibra óptica costumam ser dirigidos para o especialista, mas esse é mais acessível do que a maioria. Ele abrange guias de onda, fontes de luz, detectores de luz, acopladores, modulação, ruído e muitos outros tópicos.

Su, *The UMTS Air Interface in RF Engineering*

Oferece uma visão geral detalhada de um dos principais sistemas de celular 3G. Enfoca a interface com o ar, ou protocolos sem fios usados entre dispositivos móveis e a infraestrutura de rede.

Want, *RFID Explained*

O livro de Want é um manual de fácil leitura sobre como funciona a tecnologia incomum da camada física RFID. Ele aborda todos os aspectos da RFID, incluindo suas aplicações potenciais. Também aparecem alguns exemplos do mundo real sobre implementações de RFID e a experiência obtida com elas.

9.1.3 A camada de enlace de dados

Kasim, *Delivering Carrier Ethernet*

Atualmente, a Ethernet não é apenas uma tecnologia de área local. A nova moda é usá-la como enlace de longa distância, a Ethernet em nível global sobre diferentes tecnologias. O livro reúne ensaios que abordam o tópico em profundidade.

Lin e Costello, *Error Control Coding*, 2.ed.

Códigos para detectar e corrigir erros são fundamentais para redes de computadores confiáveis. Esse conhecido livro didático explica alguns dos códigos mais importantes, dos simples códigos lineares de Hamming aos complexos códigos de verificação de paridade de baixa densidade. Ele tenta fazer isso com o mínimo de álgebra necessário, mas ainda é bastante.

Stallings, *Data and Computer Communications*, 9.ed.

A Parte 2 aborda a transmissão de dados digitais e uma série de enlaces, inclusive detecção e controle de erros com retransmissões, e controle de fluxo.

9.1.4 A subcamada de controle de acesso ao meio

Andrews et al., *Fundamentals of WiMAX*

Livro abrangente que oferece um tratamento definitivo da tecnologia WiMAX, desde a ideia de redes sem fios em banda larga até as técnicas de redes sem fios que usam OFDM e múltiplas antenas, através do sistema multiacesso. Seu estilo em tutorial oferece o tratamento mais acessível que você encontrará para esse material pesado.

Gast, *802.11 Wireless Networks*, 2.ed.

Para uma agradável leitura sobre introdução à tecnologia e protocolos do 802.11, esse é um bom ponto de partida. Ele começa com a subcamada MAC, depois apresenta material sobre as diferentes camadas físicas e também segurança. Porém, a segunda edição não é nova o suficiente para ter tanto a dizer sobre 802.11n.

Perlman, *Interconnections*, 2.ed.

Para ver um tratamento confiável, porém divertido sobre bridges, roteadores e roteamento em geral, o livro de Perlman é a melhor opção. A autora projetou os algoritmos usados na bridge spanning tree IEEE 802; ela é uma das principais autoridades do mundo sobre vários aspectos do uso de redes.

9.1.5 A camada de rede

Comer, *Internetworking with TCP/IP*, Vol. 1, 5.ed.

Comer escreveu o trabalho definitivo sobre o conjunto de protocolos TCP/IP, agora em sua quinta edição. A maior parte da primeira metade trata do IP e protocolos relacionados na camada de rede. Os outros capítulos tratam principalmente das camadas mais altas e também merecem ser lidos.

Grayson et al., *IP Design for Mobile Networks*

As redes de telefone tradicionais e a Internet estão em rota de colisão, com redes telefônicas móveis sendo implementadas com IP em seu interior. Os autores explicam

como projetar uma rede usando os protocolos IP que dão suporte ao serviço de telefonia móvel.

Huitema, *Routing in the Internet*, 2.ed.

Se você quiser ter um conhecimento profundo sobre protocolos de roteamento, esse é um livro muito bom. Tanto algoritmos destacados (exemplos: RIP e CIDR) quanto os nem tanto assim (como OSPF, IGRP e BGP) são tratados em detalhes. Desenvolvimentos mais recentes não estão incluídos, pois é uma obra mais antiga, mas o assunto incluído é muito bem explicado.

Koodli e Perkins, *Mobile Inter-networking with IPv6*

Dois desenvolvimentos importantes na camada de rede são apresentados em um volume: IPv6 e Mobile IP. Os dois assuntos são muito bem explicados; além disso, Perkins foi uma das forças por detrás do Mobile IP.

Nucci e Papagiannaki, *Design, Measurement and Management of Large-Scale IP Networks*

Falamos bastante sobre como as redes funcionam, mas não como você projetaria, implementaria e controlaria uma se fosse um ISP. O livro preenche essa lacuna, ao examinar os métodos modernos para engenharia de tráfego e como os ISPs oferecem serviços com o uso de redes.

Perlman, *Interconnections*, 2.ed.

Nos Capítulos de 12 a 15, Perlman descreve muitas das questões envolvidas no projeto do algoritmo de roteamento de unicast e multicast, tanto para redes remotas quanto para de LANs. Porém, sem dúvida, a melhor parte é o Capítulo 18, em que a autora demonstra seus muitos anos de experiência com os protocolos de rede em um texto informativo e divertido. Trata-se de leitura obrigatória para projetistas de protocolos.

Stevens, *TCP/IP Illustrated*, Vol. 1

Os Capítulos de 3 a 10 oferecem um tratamento abrangente do IP e protocolos relacionados (ARP, RARP e ICMP), ilustrado com exemplos.

Varghese, *Network Algorithmics*

Gastamos muito tempo falando sobre como os roteadores e outros elementos da rede interagem entre si. Esse livro é diferente: trata de como os roteadores são realmente projetados para encaminhar pacotes em velocidades prodigiosas. Para quem quer entender a fundo essa e outras questões relacionadas, é o livro ideal. O autor é especialista em algoritmos inteligentes, que são usados na prática para implementar elementos de rede de alta velocidade em software e hardware.

9.1.6 A camada de transporte

Comer, *Internetworking with TCP/IP*, Vol. 1, 5.ed.

Como dissemos anteriormente, Comer escreveu o trabalho definitivo sobre o conjunto de protocolos TCP/IP. A segunda metade do livro trata do UDP e do TCP.

Farrell e Cahill, *Delay- and Disruption-Tolerant Networking*

O pequeno livro é o que deve ser lido para obter uma visão mais profunda sobre arquitetura, protocolos e aplicações em 'redes desafiadoras', ou 'challenged networks', que precisam operar sob condições difíceis de conectividade. Os autores participaram no desenvolvimento das DTNs no IETF DTN Research Group.

Stevens, *TCP/IP Illustrated*, Vol. 1

Os Capítulos de 17 a 24 oferecem um tratamento abrangente do TCP, ilustrado com exemplos.

9.1.7 A camada de aplicação

Berners-Lee et al., 'The World Wide Web'

Faça uma viagem de volta no tempo para uma perspectiva da Web e para onde ela está indo, com a pessoa que a inventou e alguns de seus colegas no CERN. O artigo focaliza a arquitetura da Web, URLs, HTTP e HTML, bem como direções futuras, e a compara com outros sistemas de informação distribuídos.

Held, *A Practical Guide to Content Delivery Networks*, 2.ed.

Oferece uma exposição prática de como funcionam as CDNs, enfatizando as considerações práticas no projeto e operação de uma CDN que funcione bem.

Hunter et al., *Beginning XML*, 4.ed.

Há muitos e muitos livros sobre HTML, XML e Web services. Esse, de 1.000 páginas, aborda a maior parte do que você provavelmente deseja saber. Ele explica não apenas como escrever XML e XHTML, mas também como desenvolver Web services que produzem e manipulam XML usando Ajax, SOAP e outras técnicas normalmente utilizadas na prática.

Krishnamurthy e Rexford, *Web Protocols and Practice*

Seria duro achar um livro mais abrangente sobre todos os aspectos da Web do que esse. Ele aborda clientes, servidores, proxies e caching, como você poderia esperar. Mas também há capítulos sobre tráfego e medições da Web, além de textos sobre a pesquisa atual e a melhoria da Web.

Simpson, *Video Over IP*, 2.ed.

O autor oferece uma visão ampla de como a tecnologia IP pode ser usada para mover vídeo pelas redes, tanto na Internet quanto em redes privativas projetadas para transportar vídeo. É interessante que esse livro é orientado para o aprendizado do profissional de vídeo a respeito das redes, e não o contrário.

Wittenburg, *Understanding Voice Over IP Technology*

Explica como funciona Voice over IP, desde o transporte de dados de áudio com protocolos IP e questões de qualidade de serviço até o SIP e o conjunto de protocolos H.323. Ele é detalhado por necessidade, dado o material, mas acessível e dividido em unidades de fácil compreensão.

9.1.8 Segurança na rede

Anderson, *Security Engineering*, 2.ed.

Apresenta uma mistura maravilhosa de técnicas de segurança, expressa em um entendimento de como as pessoas as utilizam (e utilizam mal). É um livro mais técnico do que *Secrets and Lies*, porém menos do que *Network Security* (ver mais adiante). Após uma introdução às técnicas básicas de segurança, capítulos inteiros são dedicados a diversas aplicações: bancárias, de comando e controle nuclear, de impressão de segurança, de biometria, de segurança física, de batalha eletrônica, de segurança da telecomunicação, de comércio eletrônico e de proteção de direito autoral.

Ferguson et al., *Cryptography Engineering*

Muitos livros lhe dizem como funcionam os algoritmos criptográficos populares. Esse texto conta como usar a criptografia — por que os protocolos criptográficos são criados dessa forma e como reuni-los em um sistema que atenderá a seus objetivos de segurança. Bastante compacto, sua leitura é essencial para quem projeta sistemas que dependem de criptografia.

Fridrich, *Steganography in Digital Media*

A esteganografia existe desde a Grécia antiga, onde a cera era fundida em tabletes vazios de modo que mensagens secretas pudessem ser aplicadas à madeira subjacente antes que a cera fosse reaplicada. Atualmente, vídeos, áudio e outros tipos de conteúdo na Internet oferecem diferentes transportes para mensagens secretas. Diversas técnicas modernas para esconder e encontrar informações em imagens são discutidas aqui.

Kaufman et al., *Network Security*, 2.ed.

Esse livro clássico e divertido é o primeiro lugar para procurar informações mais técnicas sobre algoritmos e protocolos de segurança de redes. Algoritmos e protocolos de chave secreta e pública, hashes de mensagem, autenticação, Kerberos, PKI, IPsec, SSL/TLS e segurança de e-mail são explicados cuidadosamente e com profundidade, e incluem muitos exemplos. O Capítulo 26, sobre o folclore da segurança, é uma verdadeira joia. Em segurança, o diabo está nos detalhes. Quem planeja criar um sistema de segurança que de fato será usado aprenderá muito com os conselhos práticos oferecidos nesse capítulo.

Schneier, *Secrets and Lies*

Se você leu *Cryptography Engineering* do começo ao fim, saberá tudo o que deve sobre algoritmos criptográficos. Se em seguida você fizer o mesmo com *Secrets and Lies* (o que leva muito menos tempo), aprenderá que a história não acaba nos algoritmos criptográficos. A maioria dos pontos fracos na segurança não se deve a algoritmos com falhas nem mesmo a chaves muito curtas, mas a falhas no ambiente de segurança. Para uma discussão não técnica e fascinante sobre segurança de computadores em sentido mais amplo, essa é uma leitura muito boa.

Skoudis e Liston, *Counter Hack Reloaded*, 2.ed.

A melhor maneira de parar um hacker é pensar como um. O livro mostra como os hackers veem uma rede e argumenta que a segurança deve ser uma função do projeto da rede inteira, e não uma reflexão posterior, baseada em uma tecnologia específica. Ele aborda quase todos os ataques comuns, incluindo os tipos como a 'engenharia social' que tiram proveito de usuários incautos, que nem sempre estão familiarizados com medidas de segurança do computador.

9.2 Referências bibliográficas

ABRAMSON, N. "Internet Access Using VSATs", *IEEE Commun. Magazine*, vol. 38, p. 60–68, jul. 2000.

AHMADI, S. "An Overview of Next-Generation Mobile WiMAX Technology", *IEEE Commun. Magazine*, vol. 47, p. 84–88, jun. 2009.

ALLMAN, M. e PAXSON, V. "On Estimating End-to-End Network Path Properties", *Proc. SIGCOMM '99 Conf.*, ACM, p. 263–274, 1999.

ANDERSON, C. *The Long Tail: Why the Future of Business is Selling Less of More*, edição revista e atualizada, Nova York: Hyperion, 2008a.

ANDERSON, R.J. *Security Engineering: A Guide to Building Dependable Distributed Systems*, 2.ed., Nova York: John Wiley & Sons, 2008b.

______. "Free Speech Online and Offline", *IEEE Computer*, vol. 25, p. 28–30, jun. 2002.

______. "The Eternity Service", *Proc. Pragocrypt Conf.*, CTU Publishing House, p. 242–252, 1996.

ANDREWS, J., GHOSH, A. e MUHAMED, R. *Fundamentals of WiMAX: Understanding Broadband Wireless Networking*, Upper Saddle River, NJ: Pearson Education, 2007.

ASTELY, D., DAHLMAN, E., FURUSKAR, A., JADING, Y., LINDSTROM, M. e PARKVALL, S. "LTE: The Evolution of Mobile Broadband", *IEEE Commun. Magazine*, vol. 47, p. 44–51, abr. 2009.

BALLARDIE, T., FRANCIS, P. e CROWCROFT, J. "Core Based Trees (CBT)", *Proc. SIGCOMM '93 Conf.*, ACM, p. 85–95, 1993.

BARAN, P. "On Distributed Communications: I. Introduction to Distributed Communication Networks", *Memorandum RM-420-PR*, Rand Corporation, ago. 1964.

BELLAMY, J. *Digital Telephony*, 3.ed., Nova York: John Wiley & Sons, 2000.

BELLMAN, R.E. *Dynamic Programming*, Princeton, NJ: Princeton University Press, 1957.

BELLOVIN, S. "The Security Flag in the IPv4 Header", RFC 3514, abr. 2003.

BELSNES, D. "Flow Control in the Packet Switching Networks", *Communications Networks*, Uxbridge, England: On-line, p. 349–361, 1975.

BENNET, C.H. e BRASSARD, G. "Quantum Cryptography: Public Key Distribution and Coin Tossing", *Int'l Conf. on Computer Systems and Signal Processing*, p. 175–179, 1984.

BERESFORD, A. e STAJANO, F. "Location Privacy in Pervasive Computing", *IEEE Pervasive Computing*, vol. 2, p. 46–55, jan. 2003.

BERGHEL, H.L. "Cyber Privacy in the New Millennium", *IEEE Computer*, vol. 34, p. 132–134, jan. 2001.

BERNERS-LEE, T., CAILLIAU, A., LOUTONEN, A., NIELSEN, H.F. e SECRET, A. "The World Wide Web", *Commun. of the ACM*, vol. 37, p. 76–82, ago. 1994.

BERTSEKAS, D. e GALLAGER, R. *Data Networks*, 2.ed., Englewood Cliffs, NJ: Prentice Hall, 1992.

BHATTI, S.N. e CROWCROFT, J. "QoS Sensitive Flows: Issues in IP Packet Handling", *IEEE Internet Computing*, vol. 4, p. 48–57, jul.-ago. 2000.

BIHAM, E. e SHAMIR, A. "Differential Fault Analysis of Secret Key Cryptosystems", *Proc. 17th Ann. Int'l Cryptology Conf.*, Berlim: Springer-Verlag LNCS 1294, p. 513–525, 1997.

BIRD, R., GOPAL, I., HERZBERG, A., JANSON, P.A., KUTTEN, S., MOLVA, R. e YUNG, M. "Systematic Design of a Family of Attack-Resistant Authentication Protocols", *IEEE J. on Selected Areas in Commun.*, vol. 11, p. 679–693, jun. 1993.

BIRRELL, A.D. e NELSON, B.J. "Implementing Remote Procedure Calls", *ACM Trans. on Computer Systems*, vol. 2, p. 39–59, fev. 1984.

BIRYUKOV, A., SHAMIR, A. e WAGNER, D. "Real Time Cryptanalysis of A5/1 on a PC", *Proc. Seventh Int'l Workshop on Fast Software Encryption*, Berlim: Springer-Verlag LNCS 1978, p. 1-8, 2000.

BLAZE, M. e BELLOVIN, S. "Tapping on My Network Door", *Commun. of the ACM*, vol. 43, p. 136, out. 2000.

BOGGS, D., MOGUL, J. e KENT, C. "Measured Capacity of an Ethernet: Myths and Reality", *Proc. SIGCOMM '88 Conf.*, ACM, p. 222–234, 1988.

BORISOV, N., GOLDBERG, I. e WAGNER, D. "Intercepting Mobile Communications: The Insecurity of 802.11", *Seventh Int'l Conf. on Mobile Computing and Networking*, ACM, p. 180–188, 2001.

BRADEN, R. "Requirements for Internet Hosts — Communication Layers", RFC 1122, out. 1989.

BRADEN, R., BORMAN, D. e PARTRIDGE, C. "Computing the Internet Checksum", RFC 1071, set. 1988.

BRANDENBURG, K. "MP3 and AAC Explained", *Proc. 17th Intl. Conf.: High-Quality Audio Coding*, Audio Engineering Society, p. 99–110, ago. 1999.

BRAY, T., PAOLI, J., SPERBERG-MCQUEEN, C., MALER, E., YERGEAU, F. e COWAN, J. "Extensible Markup Language (XML) 1.1 (Second Edition)", W3C Recommendation, set. 2006.

BRESLAU, L., CAO, P., FAN, L., PHILLIPS, G. e SHENKER, S. "Web Caching and Zipf-like Distributions: Evidence and Implications", *Proc. INFOCOM Conf.*, IEEE, p. 126–134, 1999.

BURLEIGH, S., HOOKE, A., TORGERSON, L., FALL, K., CERF, V., DURST, B., SCOTT, K. e WEISS, H. "Delay-Tolerant Networking: An Approach to Interplanetary Internet", *IEEE Commun. Magazine*, vol. 41, p. 128–136, jun. 2003.

BURNETT, S. e PAINE, S. *RSA Security's Official Guide to Cryptography*, Berkeley, CA: Osborne/McGraw-Hill, 2001.

BUSH, V. "As We May Think", *Atlantic Monthly*, vol. 176, p. 101–108, jul. 1945.

CAPETANAKIS, J.I. "Tree Algorithms for Packet Broadcast Channels", *IEEE Trans. on Information Theory*, vol. IT–5, p. 505–515, set. 1979.

CASTAGNOLI, G., BRAUER, S. e HERRMANN, M. "Optimization of Cyclic Redundancy-Check Codes with 24 and 32 Parity Bits", *IEEE Trans. on Commun.*, vol. 41, p. 883–892, jun. 1993.

CERF, V. e KAHN, R. "A Protocol for Packet Network Interconnection", *IEEE Trans. on Commun.*, vol. COM–2, p. 637–648, maio 1974.

CHANG, F., DEAN, J., GHEMAWAT, S., HSIEH, W., WALLACH, D., BURROWS, M., CHANDRA, T., FIKES, A. e GRUBER, R. "Bigtable: A Distributed Storage System for Structured Data", *Proc. OSDI 2006 Symp.*, USENIX, p. 15–29, 2006.

CHASE, J.S., GALLATIN, A.J. e YOCUM, K.G. "End System Optimizations for High-Speed TCP", *IEEE Commun. Magazine*, vol. 39, p. 68–75, abr. 2001.

CHEN, S. e NAHRSTEDT, K. "An Overview of QoS Routing for Next-Generation Networks", *IEEE Network Magazine*, vol. 12, p. 64–69, novembro/dez. 1998.

CHIU, D. e JAIN, R. "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks", *Comput. Netw. ISDN Syst.*, vol. 17, p. 1–4, jun. 1989.

CISCO. "Cisco Visual Networking Index: Forecast and Methodology, 2009–2014", Cisco Systems Inc., jun. 2010.

CLARK, D.D. "The Design Philosophy of the DARPA Internet Protocols", *Proc. SIGCOMM '88 Conf.*, ACM, p. 106–114, 1988.

______. "Window and Acknowledgement Strategy in TCP", RFC 813, jul. 1982.

CLARK, D.D., JACOBSON, V., ROMKEY, J. e SALWEN, H. "An Analysis of TCP Processing Overhead", *IEEE Commun. Magazine*, vol. 27, p. 23–29, jun. 1989.

CLARK, D.D., SHENKER, S. e ZHANG, L. "Supporting Real-Time Applications in an Integrated Services Packet Network", *Proc. SIGCOMM '92 Conf.*, ACM, p. 14–26, 1992.

CLARKE, A.C. “Extra-Terrestrial Relays”, *Wireless World*, 1945.

CLARKE, I., MILLER, S.G., HONG, T.W., SANDBERG, O. e WILEY, B.“Protecting Free Expression Online with Freenet”, *IEEE Internet Computing*, vol. 6, p. 40–49, jan.–fev. 2002.

COHEN, B. “Incentives Build Robustness in BitTorrent”, *Proc. First Workshop on Economics of Peer-to-Peer Systems*, jun. 2003.

COMER, D.E. *The Internet Book*, 4.ed., Englewood Cliffs, NJ: Prentice Hall, 2007.

______. *Internetworking with TCP/IP*, vol. 1, 5.ed., Englewood Cliffs, NJ: Prentice Hall, 2005.

CRAVER, S.A., WU, M., LIU, B., STUBBLEFIELD, A., SWARTZLANDER, B., WALLACH, D.W., DEAN, D. e FELTEN, E.W. “Reading Between the Lines: Lessons from the SDMI Challenge”, *Proc. 10th USENIX Security Symp.*, USENIX, 2001.

CROVELLA, M. e KRISHNAMURTHY, B. *Internet Measurement*, Nova York: John Wiley & Sons, 2006.

DAEMEN, J. e RIJMEN, V. *The Design of Rijndael*, Berlim: Springer-Verlag, 2002.

DALAL, Y. e METCALFE, R. “Reverse Path Forwarding of Broadcast Packets”, *Commun. of the ACM*, vol. 21, p. 1040–1048, dez. 1978.

DAVIE, B. e FARREL, A. *MPLS: Next Steps*, San Francisco: Morgan Kaufmann, 2008.

DAVIE, B. e REKHTER, Y. *MPLS Technology and Applications*, San Francisco: Morgan Kaufmann, 2000.

DAVIES, J. *Understanding IPv6*, 2.ed., Redmond, WA: Microsoft Press, 2008.

DAY, J.D. “The (Un)Revised OSI Reference Model”, *Computer Commun. Rev.*, vol. 25, p. 39–55, out. 1995.

DAY, J.D. e ZIMMERMANN, H. “The OSI Reference Model”, *Proc. of the IEEE*, vol. 71, p. 1334–1340, dez. 1983.

DECANDIA, G., HASTORIN, D., JAMPANI, M., KAKULAPATI, G., LAKSHMAN, A., PIL-CHIN, A., SIVASUBRAMANIAN, S., VOSSHALL, P. e VOGELS, W. “Dynamo: Amazon’s Highly Available Key-value Store”, *Proc. 19th Symp. on Operating Systems Prin.*, ACM, p. 205–220, dez. 2007.

DEERING, S.E. “SIP: Simple Internet Protocol”, *IEEE Network Magazine*, vol. 7, p. 16–28, maio/jun. 1993.

DEERING, S. e CHERITON, D. “Multicast Routing in Datagram Networks and Extended LANs”, *ACM Trans. on Computer Systems*, vol. 8, p. 85–110, maio 1990.

DEMERS, A., KESHAV, S. e SHENKER, S. “Analysis and Simulation of a Fair Queueing Algorithm”, *Internetwork: Research and Experience*, vol. 1, p. 3–26, set. 1990.

DENNING, D.E. e SACCO, G.M. “Timestamps in Key Distribution Protocols”, *Commun. of the ACM*, vol. 24, p. 533–536, ago. 1981.

DEVARAPALLI, V., WAKIKAWA, R., PETRESCU, A. e THUBERT, P. “Network Mobility (NEMO) Basic Support Protocol”, RFC 3963, jan. 2005.

DIFFIE, W. e HELLMAN, M.E. “Exhaustive Cryptanalysis of the NBS Data Encryption Standard”, *IEEE Computer*, vol. 10, p. 74–84, jun. 1977.

______. “New Directions in Cryptography”, *IEEE Trans. on Information Theory*, vol. IT–2, p. 644–654, nov. 1976.

DIJKSTRA, E.W. “A Note on Two Problems in Connexion with Graphs”, *Numer. Math.*, vol. 1, p. 269–271, out. 1959.

DILLEY, J., MAGGS, B., PARIKH, J., PROKOP, H., SITARAMAN, R., e WHEIL, B. “Globally Distributed Content Delivery”, *IEEE Internet Computing*, vol. 6, p. 50–58, 2002.

DINGLEDINE, R., MATHEWSON, N., SYVERSON, P. “Tor: The Second-Generation Onion Router”, *Proc. 13th USENIX Security Symp.*, USENIX, p. 303–320, ago. 2004.

DONAHOO, M. e CALVERT, K. *TCP/IP Sockets in C*, 2.ed., San Francisco: Morgan Kaufmann, 2009.

______. *TCP/IP Sockets in Java*, 2.ed., San Francisco: Morgan Kaufmann, 2008.

DONALDSON, G. e JONES, D. “Cable Television Broadband Network Architectures”, *IEEE Commun. Magazine*, vol. 39, p. 122–126, jun. 2001.

DORFMAN, R. “Detection of Defective Members of a Large Population”, *Annals Math. Statistics*, vol. 14, p. 436–440, 1943.

DUTCHER, B. *The NAT Handbook*, Nova York: John Wiley & Sons, 2001.

DUTTA-ROY, A. “An Overview of Cable Modem Technology and Market Perspectives”, *IEEE Commun. Magazine*, vol. 39, p. 81–88, jun. 2001.

EDELMAN, B., OSTROVSKY, M. e SCHWARZ, M. “Internet Advertising and the Generalized Second-Price Auction: Selling Billions of Dollars Worth of Keywords”, *American Economic Review*, vol. 97, p. 242–259, mar. 2007.

EL GAMAL, T. “A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms”, *IEEE Trans. on Information Theory*, vol. IT–1, p. 469–472, jul. 1985.

EPCGLOBAL. *EPC Radio-Frequency Identity Protocols Class--Generation – UHF RFID Protocol for Communication at 860-MHz to 960-MHz Version 1.2.0*, Bruxelas: EPCglobal Inc., out. 2008.

FALL, K. “A Delay-Tolerant Network Architecture for Challenged Internets”, *Proc. SIGCOMM 2003 Conf.*, ACM, p. 27–34, ago. 2003.

FALOUTSOS, M., FALOUTSOS, P. e FALOUTSOS, C. “On Power-Law Relationships of the Internet Topology”, *Proc. SIGCOMM ’99 Conf.*, ACM, p. 251–262, 1999.

FARRELL, S. e CAHILL, V. *Delay- and Disruption-Tolerant Networking*, Londres: Artech House, 2007.

FELLOWS, D. e JONES, D. "DOCSIS Cable Modem Technology", *IEEE Commun. Magazine*, vol. 39, p. 202–209, mar. 2001.

FENNER, B., HANDLEY, M., HOLBROOK, H. e KOUVELAS, I. "Protocol Independent Multicast-Sparse Mode (PIM-SM)", RFC 4601, ago. 2006.

FERGUSON, N., SCHNEIER, B. e KOHNO, T. *Cryptography Engineering: Design Principles and Practical Applications*, Nova York: John Wiley & Sons, 2010.

FLANAGAN, D. *JavaScript: The Definitive Guide*, 6.ed., Sebastopol, CA: O'Reilly, 2010.

FLETCHER, J. "An Arithmetic Checksum for Serial Transmissions", *IEEE Trans. on Commun.*, vol. COM–0, p. 247–252, jan. 1982.

FLOYD, S., HANDLEY, M., PADHYE, J. e WIDMER, J. "Equation-Based Congestion Control for Unicast Applications", *Proc. SIGCOMM 2000 Conf.*, ACM, p. 43–56, ago. 2000.

FLOYD, S. e JACOBSON, V. "Random Early Detection for Congestion Avoidance", *IEEE/ACM Trans. on Networking*, vol. 1, p. 397–413, ago. 1993.

FLUHRER, S., MANTIN, I. e SHAMIR, A. "Weakness in the Key Scheduling Algorithm of RC4", *Proc. Eighth Ann. Workshop on Selected Areas in Cryptography*, Berlim: Springer-Verlag LNCS 2259, p. 1–24, 2001.

FORD, B. "Structured Streams: A New Transport Abstraction", *Proc. SIGCOMM 2007 Conf.*, ACM, p. 361–372, 2007.

FORD, L.R., Jr. e FULKERSON, D.R. *Flows in Networks*, Princeton, NJ: Princeton University Press, 1962.

FORD, W. e BAUM, M.S. *Secure Electronic Commerce*, Upper Saddle River, NJ: Prentice Hall, 2000.

FORNEY, G.D. "The Viterbi Algorithm", *Proc. of the IEEE*, vol. 61, p. 268–278, mar. 1973.

FOULI, K. e MALER, M. "The Road to Carrier-Grade Ethernet", *IEEE Commun. Magazine*, vol. 47, p. S30–S38, mar. 2009.

FOX, A., GRIBBLE, S., BREWER, E. e AMIR, E. "Adapting to Network and Client Variability via On-Demand Dynamic Distillation", *SIGOPS Oper. Syst. Rev.*, vol. 30, p. 160–170, dez. 1996.

FRANCIS, P. "A Near-Term Architecture for Deploying Pip", *IEEE Network Magazine*, vol. 7, p. 30–37, maio/jun. 1993.

FRASER, A.G. "Towards a Universal Data Transport System", *IEEE J. on Selected Areas in Commun.*, vol. 5, p. 803–816, nov. 1983.

FRIDRICH, J. *Steganography in Digital Media: Principles, Algorithms, and Applications*, Cambridge: Cambridge University Press, 2009.

FULLER, V. e LI, T. "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", RFC 4632, ago. 2006.

GALLAGHER, R.G. "A Minimum Delay Routing Algorithm Using Distributed Computation", *IEEE Trans. on Commun.*, vol. COM–5, p. 73–85, jan. 1977.

________________ "Low-Density Parity Check Codes", *IRE Trans. on Information Theory*, vol. 8, p. 21–28, jan. 1962.

GARFINKEL, S., com SPAFFORD, G. *Web Security, Privacy, and Commerce*, Sebastopol, CA: O'Reilly, 2002.

GAST, M. *802.11 Wireless Networks: The Definitive Guide*, 2.ed., Sebastopol, CA: O'Reilly, 2005.

GERSHENFELD, N., KRIKORIAN, R. e COHEN, D. "The Internet of Things", *Scientific American*, vol. 291, p. 76–81, out. 2004.

GILDER, G. "Metcalfe's Law and Legacy", *Forbes ASAP*, 13 set. 1993.

GOODE, B. "Voice over Internet Protocol", *Proc. of the IEEE*, vol. 90, p. 1495–1517, set. 2002.

GORALSKI, W.J. *SONET*, 2.ed., Nova York: McGraw-Hill, 2002.

GRAYSON, M., SHATZKAMER, K. e WAINNER, S. *IP Design for Mobile Networks*, Indianapolis, IN: Cisco Press, 2009.

GROBE, K. e ELBERS, J. "PON in Adolescence: From TDMA to WDM-PON", *IEEE Commun. Magazine*, vol. 46, p. 26–34, jan. 2008.

GROSS, G., KAYCEE, M., LIN, A., MALIS, A. e STEPHENS, J. "The PPP Over AAL5", RFC 2364, jul. 1998.

HA, S., RHEE, I. e LISONG, X. "CUBIC: A New TCP-Friendly High-Speed TCP Variant", *SIGOPS Oper. Syst. Rev.*, vol. 42, p. 64–74, jun. 2008.

HAFNER, K. e LYON, M. *Where Wizards Stay Up Late*, Nova York: Simon & Schuster, 1998.

HALPERIN, D., HEYDT-BENJAMIN, T., RANSFORD, B., CLARK, S., DEFEND, B., MORGAN, W., FU, K., KOHNO, T. e MAISEL, W. "Pacemakers and Implantable Cardiac Defibrillators: Software Radio Attacks and Zero-Power Defenses", *IEEE Symp. on Security and Privacy*, p. 129––142, maio 2008.

HALPERIN, D., HU, W., SHETH, A. e WETHERALL, D. "802.11 with Multiple Antennas for Dummies", *Computer Commun. Rev.*, vol. 40, p. 19–25, jan. 2010.

HAMMING, R.W. "Error Detecting and Error Correcting Codes", *Bell System Tech. J.*, vol. 29, p. 147–160, abr. 1950.

HARTE, L., KELLOGG, S., DREHER, R. e SCHAFFNIT, T. *The Comprehensive Guide to Wireless Technology*, Fuquay-Varina, NC: APDG Publishing, 2000.

HAWLEY, G.T. "Historical Perspectives on the U.S. Telephone Loop", *IEEE Commun. Magazine*, vol. 29, p. 24–28, mar. 1991.

HECHT, J. *Understanding Fiber Optics*, Upper Saddle River, NJ: Prentice Hall, 2005.

HELD, G. *A Practical Guide to Content Delivery Networks*, 2.ed., Boca Raton, FL: CRC Press, 2010.

HEUSSE, M., ROUSSEAU, F., BERGER-SABBATEL, G., DUDA, A. "Performance Anomaly of 802.11b", *Proc. INFOCOM Conf.*, IEEE, p. 836–843, 2003.

HIERTZ, G., DENTENEER, D., STIBOR, L., ZANG, Y., COSTA, X. e WALKE, B."The IEEE 802.11 Universe", *IEEE Commun. Magazine*, vol. 48, p. 62–70, jan. 2010.

HOE, J. "Improving the Start-up Behavior of a Congestion Control Scheme for TCP", *Proc. SIGCOMM '96 Conf.*, ACM, p. 270–280, 1996.

HU, Y. e LI, V.O.K. "Satellite-Based Internet: A Tutorial", *IEEE Commun. Magazine*, vol. 30, p. 154–162, mar. 2001.

HUITEMA, C. *Routing in the Internet*, 2.ed., Englewood Cliffs, NJ: Prentice Hall, 1999.

HULL, B., BYCHKOVSKY, V., CHEN, K., GORACZKO, M., MIU, A., SHIH, E., ZHANG, Y., BALAKRISHNAN, H. e MADDEN, S. "CarTel: A Distributed Mobile Sensor Computing System", *Proc. Sensys 2006 Conf.*, ACM, p. 125–138, nov. 2006.

HUNTER, D., RAFTER, J., FAWCETT, J., VAN DER LIST, E., AYERS, D., DUCKETT, J., WATT, A. e MCKINNON, L. *Beginning XML*, 4.ed., New Jersey: Wrox, 2007.

IRMER, T. "Shaping Future Telecommunications: The Challenge of Global Standardization", *IEEE Commun. Magazine*, vol. 32, p. 20–28, jan. 1994.

ITU (INTERNATIONAL TELECOMMUNICATION UNION). *ITU Internet Reports 2005: The Internet of Things*, Genebra: ITU, nov. 2005.

______. *Measuring the Information Society: The ICT Development Index*, Geneva: ITU, mar. 2009.

JACOBSON, V. "Compressing TCP/IP Headers for Low--Speed Serial Links", RFC 1144, fev. 1990.

______. "Congestion Avoidance and Control", *Proc. SIGCOMM '88 Conf.*, ACM, p. 314–329, 1988.

JAIN, R. e ROUTHIER, S. "Packet Trains — Measurements and a New Model for Computer Network Traffic", *IEEE J. on Selected Areas in Commun.*, vol. 6, p. 986–995, set. 1986.

JAKOBSSON, M. e WETZEL, S. "Security Weaknesses in Bluetooth", *Topics in Cryptology: CT-RSA 2001*, Berlim: Springer-Verlag LNCS 2020, p. 176–191, 2001.

JOEL, A. "Telecommunications and the IEEE Communications Society", *IEEE Commun. Magazine*, 50th Anniversary Issue, p. 6–14 e 162–167, maio 2002.

JOHNSON, D., PERKINS, C. e ARKKO, J. "Mobility Support in IPv6", RFC 3775, jun. 2004.

JOHNSON, D.B., MALTZ, D. e BROCH, J. "DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks", *Ad Hoc Networking*, Boston: Addison-Wesley, p. 139–172, 2001.

JUANG, P., OKI, H., WANG, Y., MARTONOSI, M., PEH, L. e RUBENSTEIN, D. "Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet", *SIGOPS Oper. Syst. Rev.*, vol. 36, p. 96–107, out. 2002.

KAHN, D. *The Codebreakers*, 2.ed., Nova York: Macmillan, 1995.

KAMOUN, F. e KLEINROCK, L. "Stochastic Performance Evaluation of Hierarchical Routing for Large Networks", *Computer Networks*, vol. 3, p. 337–353, nov. 1979.

KARN, P. "MACA — A New Channel Access Protocol for Packet Radio", *ARRL/CRRL Amateur Radio Ninth Computer Networking Conf.*, p. 134–140, 1990.

KARN, P. e PARTRIDGE, C. "Improving Round-Trip Estimates in Reliable Transport Protocols", *Proc. SIGCOMM '87 Conf.*, ACM, p. 2–7, 1987.

KARP, B. e KUNG, H.T. "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks", *Proc. MOBICOM 2000 Conf.*, ACM, p. 243–254, 2000.

KASIM, A. *Delivering Carrier Ethernet*, Nova York: McGraw--Hill, 2007.

KATABI, D., HANDLEY, M. e ROHRS, C. "Internet Congestion Control for Future High Bandwidth-Delay Product Environments", *Proc. SIGCOMM 2002 Conf.*, ACM, p. 89–102, 2002.

KATZ, D. e FORD, P.S. "TUBA: Replacing IP with CLNP", *IEEE Network Magazine*, vol. 7, p. 38–47, maio/jun. 1993.

KAUFMAN, C., PERLMAN, R. e SPECINER, M. *Network Security*, 2.ed., Englewood Cliffs, NJ: Prentice Hall, 2002.

KENT, C. e MOGUL, J. "Fragmentation Considered Harmful", *Proc. SIGCOMM '87 Conf.*, ACM, p. 390–401, 1987.

KERCKHOFF, A. "La Cryptographie Militaire", *J. des Sciences Militaires*, vol. 9, p. 5–38, jan. 1883 e p. 161–191, fev. 1883.

KHANNA, A. e ZINKY, J. "The Revised ARPANET Routing Metric", *Proc. SIGCOMM '89 Conf.*, ACM, p. 45–56, 1989.

KIPNIS, J. "Beating the System: Abuses of the Standards Adoption Process", *IEEE Commun. Magazine*, vol. 38, p. 102–105, jul. 2000.

KLEINROCK, L. "Power and Other Deterministic Rules of Thumb for Probabilistic Problems in Computer Communications", *Proc. Intl. Conf. on Commun.*, p. 43.1.1–43.1.10, jun. 1979.

KLEINROCK, L. e TOBAGI, F. "Random Access Techniques for Data Transmission over Packet-Switched Radio Channels", *Proc. Nat. Computer Conf.*, p. 187–201, 1975.

KOHLER, E., HANDLEY, H. e FLOYD, S. "Designing DCCP: Congestion Control without Reliability", *Proc. SIGCOMM 2006 Conf.*, ACM, p. 27–38, 2006.

KOODLI, R. e PERKINS, C.E. *Mobile Inter-networking with IPv6*, Nova York: John Wiley & Sons, 2007.

KOOPMAN, P. "32-Bit Cyclic Redundancy Codes for Internet Applications", *Proc. Intl. Conf. on Dependable Systems and Networks.*, IEEE, p. 459–472, 2002.

KRISHNAMURTHY, B. e REXFORD, J. *Web Protocols and Practice*, Boston: Addison-Wesley, 2001.

KUMAR, S., PAAR, C., PELZL, J. PFEIFFER, G. e SCHIMMLER, M. "Breaking Ciphers with COPACOBANA: A Cost-Optimized Parallel Code Breaker", *Proc. 8th Cryp-*

tographic Hardware and Embedded Systems Wksp., IACR, p. 101–118, out. 2006.

LABOVITZ, C., AHUJA, A., BOSE, A. e JAHANIAN, F. "Delayed Internet Routing Convergence", *IEEE/ACM Trans. on Networking*, vol. 9, p. 293–306, jun. 2001.

LAM, C.K.M. e TAN, B.C.Y. "The Internet Is Changing the Music Industry", *Commun. of the ACM*, vol. 44, p. 62–66, ago. 2001.

LAOUTARIS, N., SMARAGDAKIS, G., RODRIGUEZ, P. e SUNDARAM, R. "Delay Tolerant Bulk Data Transfers on the Internet", *Proc. SIGMETRICS 2009 Conf.*, ACM, p. 229–238, jun. 2009.

LARMO, A., LINDSTROM, M., MEYER, M., PELLETIER, G., TORSNER, J. e WIEMANN, H. "The LTE Link-Layer Design", *IEEE Commun. Magazine*, vol. 47, p. 52–59, abr. 2009.

LEE, J.S. e MILLER, L.E. *CDMA Systems Engineering Handbook*, Londres: Artech House, 1998.

LELAND, W., TAQQU, M., WILLINGER, W. e WILSON, D. "On the Self-Similar Nature of Ethernet Traffic", *IEEE/ACM Trans. on Networking*, vol. 2, p. 1–15, fev. 1994.

LEMON, J. "Resisting SYN Flood DOS Attacks with a SYN Cache", *Proc. BSDCon Conf.*, USENIX, p. 88–98, 2002.

LEVY, S. "Crypto Rebels", *Wired*, p. 54–61, maio/jun. 1993.

LEWIS, M. *Comparing, Designing, and Deploying VPNs*, Indianapolis, IN: Cisco Press, 2006.

LI, M., AGRAWAL, D., GANESAN, D. e VENKATARAMANI, A. "Block-Switched Networks: A New Paradigm for Wireless Transport", *Proc. NSDI 2009 Conf.*, USENIX, p. 423–436, 2009.

LIN, S. e COSTELLO, D. *Error Control Coding*, 2.ed., Upper Saddle River, NJ: Pearson Education, 2004.

LUBACZ, J., MAZURCZYK, W. e SZCZYPIORSKI, K. "Vice over IP", *IEEE Spectrum*, p. 42–47, fev. 2010.

MACEDONIA, M.R. "Distributed File Sharing", *IEEE Computer*, vol. 33, p. 99–101, 2000.

MADHAVAN, J., KO, D., LOT, L., GANGPATHY, V., RASMUSSEN, A. e HALEVY, A. "Google's Deep Web Crawl", *Proc. VLDB 2008 Conf.*, VLDB Endowment, p. 1241–1252, 2008.

MAHAJAN, R., RODRIG, M., WETHERALL, D. e ZAHORJAN, J. "Analyzing the MAC-Level Behavior of Wireless Networks in the Wild", *Proc. SIGCOMM 2006 Conf.*, ACM, p. 75–86, 2006.

MALIS, A. e SIMPSON, W. "PPP over SONET/SDH", RFC 2615, jun. 1999.

MASSEY, J.L. 'Shift-Register Synthesis and BCH Decoding', *IEEE Trans. on Information Theory*, vol. IT–5, p. 122–127, jan. 1969.

MATSUI, M. "Linear Cryptanalysis Method for DES Cipher", *Advances in Cryptology — Eurocrypt 1993 Proceedings*, Berlim: Springer-Verlag LNCS 765, p. 386–397, 1994.

MAUFER, T.A. *IP Fundamentals*, Upper Saddle River, NJ: Prentice Hall, 1999.

MAYMOUNKOV, P. e MAZIERES, D. "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric", *Proc. First Intl. Wksp. on Peer-to-Peer Systems*, Berlim: Springer-Verlag LNCS 2429, p. 53–65, 2002.

MAZIERES, D. e KAASHOEK, M.F. "The Design, Implementation, and Operation of an Email Pseudonym Server", *Proc. Fifth Conf. on Computer and Commun. Security*, ACM, p. 27–36, 1998.

MCAFEE LABS. *McAfee Threat Reports: First Quarter 2010*, McAfee Inc., 2010.

MENEZES, A.J. e VANSTONE, S.A. "Elliptic Curve Cryptosystems and Their Implementation", *Journal of Cryptology*, vol. 6, p. 209–224, 1993.

MERKLE, R.C. e HELLMAN, M. "Hiding and Signatures in Trapdoor Knapsacks", *IEEE Trans. on Information Theory*, vol. IT–4, p. 525–530, set. 1978.

METCALFE, R.M. "Computer/Network Interface Design: Lessons from Arpanet and Ethernet", *IEEE J. on Selected Areas in Commun.*, vol. 11, p. 173–179, fev. 1993.

METCALFE, R.M. e BOGGS, D.R. "Ethernet: Distributed Packet Switching for Local Computer Networks", *Commun. of the ACM*, vol. 19, p. 395–404, jul. 1976.

METZ, C. "Interconnecting ISP Networks", *IEEE Internet Computing*, vol. 5, p. 74–80, mar.–abr. 2001.

MISHRA, P.P., KANAKIA, H. e TRIPATHI, S. "On Hop by Hop Rate-Based Congestion Control", *IEEE/ACM Trans. on Networking*, vol. 4, p. 224–239, abr. 1996.

MOGUL, J.C. "IP Network Performance", em *Internet System Handbook*, D.C. Lynch e M.Y. Rose (eds.), Boston: Addison-Wesley, p. 575–575, 1993.

MOGUL, J. e DEERING, S. "Path MTU Discovery", RFC 1191, nov. 1990.

MOGUL, J. e MINSHALL, G. "Rethinking the Nagle Algorithm", *Comput. Commun. Rev.*, vol. 31, p. 6–20, jan. 2001.

MOY, J. "Multicast Routing Extensions for OSPF", *Commun. of the ACM*, vol. 37, p. 61–66, ago. 1994.

MULLINS, J. "Making Unbreakable Code", *IEEE Spectrum*, p. 40–45, maio 2002.

NAGLE, J. "On Packet Switches with Infinite Storage", *IEEE Trans. on Commun.*, vol. COM–5, p. 435–438, abr. 1987.

_________. "Congestion Control in TCP/IP Internetworks", *Computer Commun. Rev.*, vol. 14, p. 11–17, out. 1984.

NAUGHTON, J. *A Brief History of the Future*, Woodstock, NY: Overlook Press, 2000.

NEEDHAM, R.M. e SCHROEDER, M.D. "Using Encryption for Authentication in Large Networks of Computers", *Commun. of the ACM*, vol. 21, p. 993–999, dez. 1978.

_________. "Authentication Revisited", *Operating Systems Rev.*, vol. 21, p. 7, jan. 1987.

NELAKUDITI, S. e ZHANG, Z.-L. “A Localized Adaptive Proportioning Approach to QoS Routing”, *IEEE Commun. Mag.*, vol. 40, p. 66–71, jun. 2002.

NEUMAN, C. e TS’O, T. “Kerberos: An Authentication Service for Computer Networks”, *IEEE Commun. Mag.*, vol. 32, p. 33–38, set. 1994.

NICHOLS, R.K. e LEKKAS, P.C. *Wireless Security*, Nova York: McGraw-Hill, 2002.

NIST. “Secure Hash Algorithm”, U.S. Government Federal Information Processing Standard 180, 1993.

NONNENMACHER, J., BIERSACK, E. e TOWSLEY, D. “Parity-Based Loss Recovery for Reliable Multicast Transmission”, *Proc. SIGCOMM ’97 Conf.*, ACM, p. 289–300, 1997.

NUCCI, A. e PAPAGIANNAKI, D. *Design, Measurement and Management of Large-Scale IP Networks*, Cambridge: Cambridge University Press, 2008.

NUGENT, R., MUNAKANA, R., CHIN, A., COELHO, R. e PUIG-SUARI, J. “The CubeSat: The PicoSatellite Standard for Research and Education”, *Proc. SPACE 2008 Conf.*, AIAA, 2008.

ORAN, D. "OSI IS-IS Intra-domain Routing Protocol", RFC 1142, fev. 1990.

OTWAY, D. e REES, O. “Efficient and Timely Mutual Authentication”, *Operating Systems Rev.*, p. 8–10, jan. 1987.

PADHYE, J., FIROIU, V., TOWSLEY, D. e KUROSE, J. “Modeling TCP Throughput: A Simple Model and Its Empirical Validation”, *Proc. SIGCOMM ’98 Conf.*, ACM, p. 303–314, 1998.

PALAIS, J.C. *Fiber Optic Commun.*, 5.ed., Englewood Cliffs, NJ: Prentice Hall, 2004.

PARAMESWARAN, M., SUSARLA, A. e WHINSTON, A.B. “P2P Networking: An Information-Sharing Alternative”, *IEEE Computer*, vol. 34, p. 31–38, jul. 2001.

PAREKH, A. e GALLAGHER, R. “A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Multiple-Node Case”, *IEEE/ACM Trans. on Networking*, vol. 2, p. 137–150, abr. 1994.

_________. “A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case”, *IEEE/ACM Trans. on Networking*, vol. 1, p. 344–357, jun. 1993.

PARTRIDGE, C., HUGHES, J. e STONE, J. "Performance of Checksums and CRCs over Real Data", *Proc. SIGCOMM ’95 Conf.*, ACM, p. 68–76, 1995.

PARTRIDGE, C., MENDEZ, T. e MILLIKEN, W. "Host Anycasting Service", RFC 1546, nov. 1993.

PAXSON, V. e FLOYD, S. "Wide-Area Traffic: The Failure of Poisson Modeling", *IEEE/ACM Trans. on Networking*, vol. 3, p. 226–244, jun. 1995.

PERKINS, C. "IP Mobility Support for IPv4", RFC 3344, ago. 2002.

PERKINS, C.E. *RTP: Audio and Video for the Internet*, Boston: Addison-Wesley, 2003.

_________. (ed.). *Ad Hoc Networking*, Boston: Addison-Wesley, 2001.

_________. *Mobile IP Design Principles and Practices*, Upper Saddle River, NJ: Prentice Hall, 1998.

PERKINS, C.E. e ROYER, E. “The Ad Hoc On-Demand Distance-Vector Protocol”, em *Ad Hoc Networking*, editado por C. Perkins, Boston: Addison-Wesley, 2001.

PERLMAN, R. *Interconnections*, 2.ed., Boston: Addison-Wesley, 2000.

_________. *Network Layer Protocols with Byzantine Robustness*, tese de Ph.D., M.I.T., 1988.

_________. “An Algorithm for the Distributed Computation of a Spanning Tree in an Extended LAN”, *Proc. SIGCOMM ’85 Conf.*, ACM, p. 44–53, 1985.

PERLMAN, R. e KAUFMAN, C. “Key Exchange in IPsec”, *IEEE Internet Computing*, vol. 4, p. 50–56, nov.–dez. 2000.

PETERSON, W.W. e BROWN, D.T. “Cyclic Codes for Error Detection”, *Proc. IRE*, vol. 49, p. 228–235, jan. 1961.

PIATEK, M., KOHNO, T. e KRISHNAMURTHY, A. “Challenges and Directions for Monitoring P2P File Sharing Networks — or Why My Printer Received a DMCA Takedown Notice”, *3rd Workshop on Hot Topics in Security*, USENIX, jul. 2008.

PIATEK, M., ISDAL, T., ANDERSON, T., KRISHNAMURTHY, A. e VENKA-TARAMANI, V. “Do Incentives Build Robustness in BitTorrent?”, *Proc. NSDI 2007 Conf.*, USENIX, p. 1–14, 2007.

PISCITELLO, D.M. e CHAPIN, A.L. *Open Systems Networking: TCP/IP and OSI*, Boston: Addison-Wesley, 1993.

PIVA, A., BARTOLINI, F. e BARNI, M. “Managing Copyrights in Open Networks”, *IEEE Internet Computing*, vol. 6, p. 18–26, maio 2002.

POSTEL, J. “Internet Control Message Protocols”, RFC 792, set. 1981.

RABIN, J. e MCCATHIENEVILE, C. “Mobile Web Best Practices 1.0”, W3C Recommendation, jul. 2008.

RAMAKRISHNAM, K.K., FLOYD, S. e BLACK, D. “The Addition of Explicit Congestion Notification (ECN) to IP”, RFC 3168, set. 2001.

RAMAKRISHNAN, K.K. e JAIN, R. “A Binary Feedback Scheme for Congestion Avoidance in Computer Networks with a Connectionless Network Layer”, *Proc. SIGCOMM ’88 Conf.*, ACM, p. 303–313, 1988.

RAMASWAMI, R., KUMAR, S. e SASAKI, G. *Optical Networks: A Practical Perspective*, 3.ed., San Francisco: Morgan Kaufmann, 2009.

RATNASAMY, S., FRANCIS, P., HANDLEY, M., KARP, R. e SHENKER, S. “A Scalable Content-Addressable Network”, *Proc. SIGCOMM 2001 Conf.*, ACM, p. 161–172, 2001.

RIEBACK, M., CRISPO, B. e TANENBAUM, A. "Is Your Cat Infected with a Computer Virus?", *Proc. IEEE Percom*, p. 169–179, mar. 2006.

RIVEST, R.L. "The MD5 Message-Digest Algorithm", RFC 1320, abr. 1992.

RIVEST, R.L., SHAMIR, A. e ADLEMAN, L. "On a Method for Obtaining Digital Signatures and Public Key Cryptosystems", *Commun. of the ACM*, vol. 21, p. 120–126, fev. 1978.

ROBERTS, L.G. "Extensions of Packet Communication Technology to a Hand Held Personal Terminal", *Proc. Spring Joint Computer Conf.*, AFIPS, p. 295–298, 1972.

_________. "Multiple Computer Networks and Intercomputer Communication", *Proc. First Symp. on Operating Systems Prin.*, ACM, p. 3.1–3.6, 1967.

ROSE, M.T. *The Simple Book*, Englewood Cliffs, NJ: Prentice Hall, 1994.

_________. *The Internet Message*, Englewood Cliffs, NJ: Prentice Hall, 1993.

ROWSTRON, A. e DRUSCHEL, P. "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Storage Utility", *Proc. 18th Int'l Conf. on Distributed Systems Platforms*, Londres: Springer-Verlag LNCS 2218, p. 329–350, 2001.

RUIZ-SANCHEZ, M.A., BIERSACK, E.W. e DABBOUS, W. "Survey and Taxonomy of IP Address Lookup Algorithms", *IEEE Network Magazine*, vol. 15, p. 8–23, mar.–abr. 2001.

SALTZER, J.H., REED, D.P. e CLARK, D.D. "End-to-End Arguments in System Design", *ACM Trans. on Computer Systems*, vol. 2, p. 277–288, nov. 1984.

SAMPLE, A., YEAGER, D., POWLEDGE, P., MAMISHEV, A. e SMITH, J. "Design of an RFID-Based Battery-Free Programmable Sensing Platform", *IEEE Trans. on Instrumentation and Measurement*, vol. 57, p. 2608–2615, nov. 2008.

SAROIU, S., GUMMADI, K. e GRIBBLE, S. "Measuring and Analyzing the Characteristics of Napster & Gnutella Hosts", *Multim. Syst.*, vol. 9, p. 170–184, ago. 2003.

SCHALLER, R. "Moore's Law: Past, Present and Future", *IEEE Spectrum*, vol. 34, p. 52–59, jun. 1997.

SCHNEIER, B. *Secrets and Lies*, Nova York: John Wiley & Sons, 2004.

_________. *E-Mail Security*, Nova York: John Wiley & Sons, 1995.

SCHNORR, C.P. "Efficient Signature Generation for Smart Cards", *Journal of Cryptology*, vol. 4, p. 161–174, 1991.

SCHOLTZ, R.A. "The Origins of Spread-Spectrum Communications", *IEEE Trans. on Commun.*, vol. COM–0, p. 822–854, maio 1982.

SCHWARTZ, M. e ABRAMSON, N. "The AlohaNet: Surfing for Wireless Data", *IEEE Commun. Magazine*, vol. 47, p. 21–25, dez. 2009.

SEIFERT, R. e EDWARDS, J. *The All-New Switch Book*, NY: John Wiley, 2008.

SENN, J.A. "The Emergence of M-Commerce", *IEEE Computer*, vol. 33, p. 148–150, dez. 2000.

SERJANTOV, A. "Anonymizing Censorship Resistant Systems", *Proc. First Int'l Workshop on Peer-to-Peer Systems*, Londres: Springer-Verlag LNCS 2429, p. 111–120, 2002.

SHACHAM, N. e MCKENNY, P. "Packet Recovery in High-Speed Networks Using Coding and Buffer Management", *Proc. INFOCOM Conf.*, IEEE, p. 124–131, jun. 1990.

SHAIKH, A., REXFORD, J. e SHIN, K. "Load-Sensitive Routing of Long-Lived IP Flows", *Proc. SIGCOMM '99 Conf.*, ACM, p. 215–226, set. 1999.

SHALUNOV, S. e CARLSON, R. "Detecting Duplex Mismatch on Ethernet", *Passive and Active Network Measurement*, Berlim: Springer-Verlag LNCS 3431, p. 3135–3148, 2005.

SHANNON, C. "A Mathematical Theory of Communication", *Bell System Tech. J.*, vol. 27, p. 379–423, jul. 1948; e p. 623–656, out. 1948.

SHEPARD, S. *SONET/SDH Demystified*, Nova York: McGraw-Hill, 2001.

SHREEDHAR, M. e VARGHESE, G. "Efficient Fair Queueing Using Deficit Round Robin", *Proc. SIGCOMM '95 Conf.*, ACM, p. 231–243, 1995.

SIMPSON, W. *Video Over IP*, 2.ed., Burlington, MA: Focal Press, 2008.

_________. "The Point-to-Point Protocol (PPP)", RFC 1661, jul. 1994a.

_________. "PPP in HDLC-like Framing", RFC 1662, jul. 1994b.

SIU, K. e JAIN, R. "A Brief Overview of ATM: Protocol Layers, LAN Emulation, and Traffic", *ACM Computer Communications Review*, vol. 25, p. 6–20, abr. 1995.

SKOUDIS, E. e LISTON, T. *Counter Hack Reloaded*, 2.ed., Upper Saddle River, NJ: Prentice Hall, 2006.

SMITH, D.K. e ALEXANDER, R.C. *Fumbling the Future*, Nova York: William Morrow, 1988.

SNOEREN, A.C. e BALAKRISHNAN, H. "An End-to-End Approach to Host Mobility", *Int'l Conf. on Mobile Computing and Networking* , ACM, p. 155–166, 2000.

SOBEL, D.L. "Will Carnivore Devour Online Privacy", *IEEE Computer*, vol. 34, p. 87–88, maio 2001.

SOTIROV, A., STEVENS, M., APPELBAUM, J., LENSTRA, A., MOLNAR, D., OSVIK, D. e DE WEGER, B. "MD5 Considered Harmful Today", *Proc. 25th Chaos Communication Congress*, Verlag Art d'Ameublement, 2008.

SOUTHEY, R. *The Doctors*, Londres: Longman, Brown, Green and Longmans, 1848.

SPURGEON, C.E. *Ethernet: The Definitive Guide*, Sebastopol, CA: O'Reilly, 2000.

STALLINGS, W. *Data and Computer Communications*, 9.ed., Upper Saddle River, NJ: Pearson Education, 2010.

STARR, T., SORBARA, M., COIFFI, J. e SILVERMAN, P. "DSL Advances", Upper Saddle River, NJ: Prentice Hall, 2003.

STEVENS, W.R. *TCP/IP Illustrated: The Protocols*, Boston: Addison Wesley, 1994.

STINSON, D.R. *Cryptography Theory and Practice*, 2.ed., Boca Raton, FL: CRC Press, 2002.

STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, M.F. e BALAKRISHNAN, H. "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications", *Proc. SIGCOMM 2001 Conf.*, ACM, p. 149–160, 2001.

STUBBLEFIELD, A., IOANNIDIS, J. e RUBIN, A.D. "Using the Fluhrer, Mantin, and Shamir Attack to Break WEP", *Proc. Network and Distributed Systems Security Symp.*, ISOC, p. 1–11, 2002.

STUTTARD, D. e PINTO, M. *The Web Application Hacker's Handbook*, Nova York: John Wiley & Sons, 2007.

SU, S. *The UMTS Air Interface in RF Engineering*, Nova York: McGraw-Hill, 2007.

SULLIVAN, G. e WIEGAND, T. "Tree Algorithms for Packet Broadcast Channels", *Proc. of the IEEE*, vol. 93, p. 18–31, jan. 2005.

SUNSHINE, C.A. e DALAL, Y.K. "Connection Management in Transport Protocols", *Computer Networks*, vol. 2, p. 454–473, 1978.

TAN, K., SONG, J., ZHANG, Q. e SRIDHARN, M. "A Compound TCP Approach for High-Speed and Long Distance Networks", *Proc. INFOCOM Conf.*, IEEE, p. 1–12, 2006.

TANENBAUM, A.S. *Modern Operating Systems*, 3.ed., Upper Saddle River, NJ: Prentice Hall, 2007.

TANENBAUM, A.S. e VAN STEEN, M. *Distributed Systems: Principles and Paradigms*, Upper Saddle River, NJ: Prentice Hall, 2007.

TOMLINSON, R.S. "Selecting Sequence Numbers", *Proc. SIGCOMM/SIGOPS Interprocess Commun. Workshop*, ACM, p. 11–23, 1975.

TUCHMAN, W. "Hellman Presents No Shortcut Solutions to DES", *IEEE Spectrum*, vol. 16, p. 40–41, jul. 1979.

TURNER, J.S. "New Directions in Communications (or Which Way to the Information Age)", *IEEE Commun. Magazine*, vol. 24, p. 8–15, out. 1986.

UNGERBOECK, G. "Trellis-Coded Modulation with Redundant Signal Sets Part I: Introduction", *IEEE Commun. Magazine*, vol. 25, p. 5–11, fev. 1987.

VALADE, J. *PHP* & *MySQL for Dummies*, 5.ed., Nova York: John Wiley & Sons, 2009.

VARGHESE, G. *Network Algorithmics*, San Francisco: Morgan Kaufmann, 2004.

VARGHESE, G. e LAUCK, T. "Hashed and Hierarchical Timing Wheels: Data Structures for the Efficient Implementation of a Timer Facility", *Proc. 11th Symp. on Operating Systems Prin.*, ACM, p. 25–38, 1987.

VERIZON BUSINESS. *2009 Data Breach Investigations Report*, Verizon, 2009.

VITERBI, A. *CDMA: Principles of Spread Spectrum Communication*, Englewood Cliffs, NJ: Prentice Hall, 1995.

VON AHN, L., BLUM, B. e LANGFORD, J. "Telling Humans and Computers Apart Automatically", *Commun. of the ACM*, vol. 47, p. 56–60, fev. 2004.

WAITZMAN, D., PARTRIDGE, C. e DEERING, S. "Distance Vector Multicast Routing Protocol", RFC 1075, nov. 1988.

WALDMAN, M., RUBIN, A.D. e CRANOR, L.F. "Publius: A Robust, Tamper-Evident, Censorship-Resistant Web Publishing System", *Proc. Ninth USENIX Security Symp.*, USENIX, p. 59–72, 2000.

WANG, Z. e CROWCROFT, J. "SEAL Detects Cell Misordering", *IEEE Network Magazine*, vol. 6, p. 8–9, jul. 1992.

WANT, R. *RFID Explained*, San Rafael, CA: Morgan Claypool, 2006.

WARNEKE, B., LAST, M., LIEBOWITZ, B. e PISTER, K.S.J. "Smart Dust: Communicating with a Cubic Millimeter Computer", *IEEE Computer*, vol. 34, p. 44–51, jan. 2001.

WAYNER, P. *Disappearing Cryptography: Information Hiding, Steganography, and Watermarking*, 3.ed., San Francisco: Morgan Kaufmann, 2008.

WEI, D., CHENG, J., LOW, S. e HEGDE, S. "FAST TCP: Motivation, Architecture, Algorithms, Performance", *IEEE/ACM Trans. on Networking*, vol. 14, p. 1246–1259, dez. 2006.

WEISER, M. "The Computer for the Twenty-First Century", *Scientific American*, vol. 265, p. 94–104, set. 1991.

WELBOURNE, E., BATTLE, L., COLE, G., GOULD, K., RECTOR, K., RAYMER, S., BALAZINSKA, M. e BORRIELLO, G. "Building the Internet of Things Using RFID", *IEEE Internet Computing*, vol. 13, p. 48–55, maio 2009.

WITTENBURG, N. *Understanding Voice Over IP Technology*, Clifton Park, NY: Delmar Cengage Learning, 2009.

WOLMAN, A., VOELKER, G., SHARMA, N., CARDWELL, N., KARLIN, A. e LEVY, H. "On the Scale and Performance of Cooperative Web Proxy Caching", *Proc. 17th Symp. on Operating Systems Prin.*, ACM, p. 16–31, 1999.

WOOD, L., IVANCIC, W., EDDY, W., STEWART, D., NORTHAM, J., JACKSON, C. e DA SILVA CURIEL, A. "Use of the Delay-Tolerant Networking Bundle Protocol from Space", *Proc. 59th Int'l Astronautical Congress*, Int'l Astronautical Federation, p. 3123–3133, 2008.

WU, T. "Network Neutrality, Broadband Discrimination", *Journal on Telecom. and High-Tech. Law*, vol. 2, p. 141–179, 2003.

WYLIE, J., BIGRIGG, M.W., STRUNK, J.D., GANGER, G.R., KILICCOTE, H. e KHOSLA, P.K. "Survivable Informa-

tion Storage Systems", *IEEE Computer*, vol. 33, p. 61–68, ago. 2000.

YU, T., HARTMAN, S. e RAEBURN, K. "The Perils of Unauthenticated Encryption: Kerberos Version 4", *Proc. NDSS Symposium*, Internet Society, fev. 2004.

YUVAL, G. "How to Swindle Rabin", *Cryptologia*, vol. 3, p. 187–190, jul. 1979.

ZACKS, M. "Antiterrorist Legislation Expands Electronic Snooping", *IEEE Internet Computing*, vol. 5, p. 8–9, nov.–dez. 2001.

ZHANG, Y., BRESLAU, L., PAXSON, V. e SHENKER, S. "On the Characteristics and Origins of Internet Flow Rates", *Proc. SIGCOMM 2002 Conf.*, ACM, p. 309–322, 2002.

ZHAO, B., LING, H., STRIBLING, J., RHEA, S., JOSEPH, A. e KUBIATOWICZ, J. "Tapestry: A Resilient Global-Scale Overlay for Service Deployment", *IEEE J. on Selected Areas in Commun.*, vol. 22, p. 41–53, jan. 2004.

ZIMMERMANN, P.R. *The Official PGP User's Guide*, Cambridge, MA: M.I.T. Press, 1995a.

_________. *PGP: Source Code and Internals*, Cambridge, MA: M.I.T. Press, 1995b.

ZIPF, G.K. *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*, Boston: Addison-Wesley, 1949.

ZIV, J. e LEMPEL, Z. "A Universal Algorithm for Sequential Data Compression", *IEEE Trans. on Information Theory*, vol. IT–3, p. 337–343, maio 1977.

Índice remissivo

Números

A

B

C

D

E

F

G

H

I

N

O

P

Q

R

S

T

U

V

W

X

Z

Sobre os autores

Andrew S. Tanenbaum é bacharel em ciências pelo MIT e Ph.D. pela Universidade da Califórnia em Berkeley. É professor de ciência da computação na Universidade Vrije em Amsterdã, Holanda, onde leciona sistemas operacionais, redes e assuntos relacionados há mais de 30 anos. Desenvolve pesquisas sobre sistemas operacionais altamente confiáveis, embora tenha trabalhado com compiladores, sistemas distribuídos, segurança e outros tópicos no decorrer dos anos. Esses projetos de pesquisa resultaram em mais de 150 artigos em periódicos e conferências.

O professor Tanenbaum também foi autor e coautor de cinco livros que até agora já tiveram 19 edições. Os livros foram traduzidos para 21 idiomas, do basco ao tailandês, e são usados em universidades no mundo inteiro. Ao todo, existem 159 versões (combinações de idioma/edição), que estão listadas em www.cs.vu.nl/~ast/publications.

O autor também já produziu um considerável volume de softwares, incluindo: Amsterdam Compiler Kit (um kit de ferramentas para o desenvolvimento de compiladores portáteis); Amoeba (um antigo sistema distribuído usado em redes locais); e Globe (um sistema distribuído remoto).

É também o autor do MINIX, um pequeno clone do UNIX, destinado inicialmente ao uso em laboratórios de programação para alunos, e que foi a inspiração direta para o Linux e sua plataforma inicial. A versão atual, chamada MINIX 3, tem sido considerada um sistema operacional extremamente confiável e seguro. O professor considerará seu trabalho encerrado quando nenhum computador for equipado com um botão de 'reset' e ninguém mais tiver experimentado uma falha de sistema. O MINIX 3 é um projeto de código aberto em andamento, para o qual você está convidado a contribuir. Visite www.minix3.org para baixar uma cópia gratuita e descobrir o que está acontecendo.

Tanenbaum é membro do conselho da ACM, membro do conselho do IEEE e membro da Royal Netherlands Academy of Arts and Sciences. Também ganhou diversos prêmios científicos, entre os quais:

- 2010 TAA McGuffey Award por livros de ciência da computação e engenharia;
- 2007 IEEE James H. Mulligan, Jr. Education Medal;
- 2002 TAA Texty Award por livros de ciência da computação e engenharia;
- 1997 ACM/SIGCSE Award por contribuições relevantes para o ensino da ciência da computação;
- 1994 ACM Karl V. Karlstrom Outstanding Educator Award.

Sua home page pode ser encontrada em http://www.cs.vu.nl/~ast/.

David J. Wetherall é professor adjunto de ciência da computação e engenharia na Universidade de Washington e consultor da Intel Labs, ambas em Seattle. Obteve o bacharelado em engenharia elétrica pela Universidade do Oeste Australiano e o Ph.D. em ciência da computação pelo MIT.

O autor trabalhou na área de redes durante as duas últimas décadas. Sua pesquisa está focada em sistemas de rede, especialmente redes sem fios e computação móvel, projeto de protocolos da Internet e medição de rede.

Wetherall recebeu o prêmio ACM SIGCOMM Test-of-Time pela pesquisa pioneira em redes ativas, uma arquitetura para introduzir novos serviços de rede com rapidez. Recebeu também o IEEE William Bennett Prize por inovações no mapeamento da Internet. Sua pesquisa foi reconhecida com um prêmio NSF CAREER em 2002, e ele se tornou Sloan Fellow em 2004.

Além de lecionar redes, o professor Wetherall participa da comunidade de pesquisa sobre essa disciplina. Ele também é copresidente dos comitês de programa do SIGCOMM, NSDI e MobiSys, e cofundador dos workshops ACM HotNets. Vem auxiliando diversos comitês de programa para conferências sobre redes e é editor da *ACM Computer Communication Review*.